▼ AMSTRAD ▼ SPECTRUM ▼ COMMODORE ▼ IBM ▼ MSX ▼

Informática y programación

20

PASO A PASO



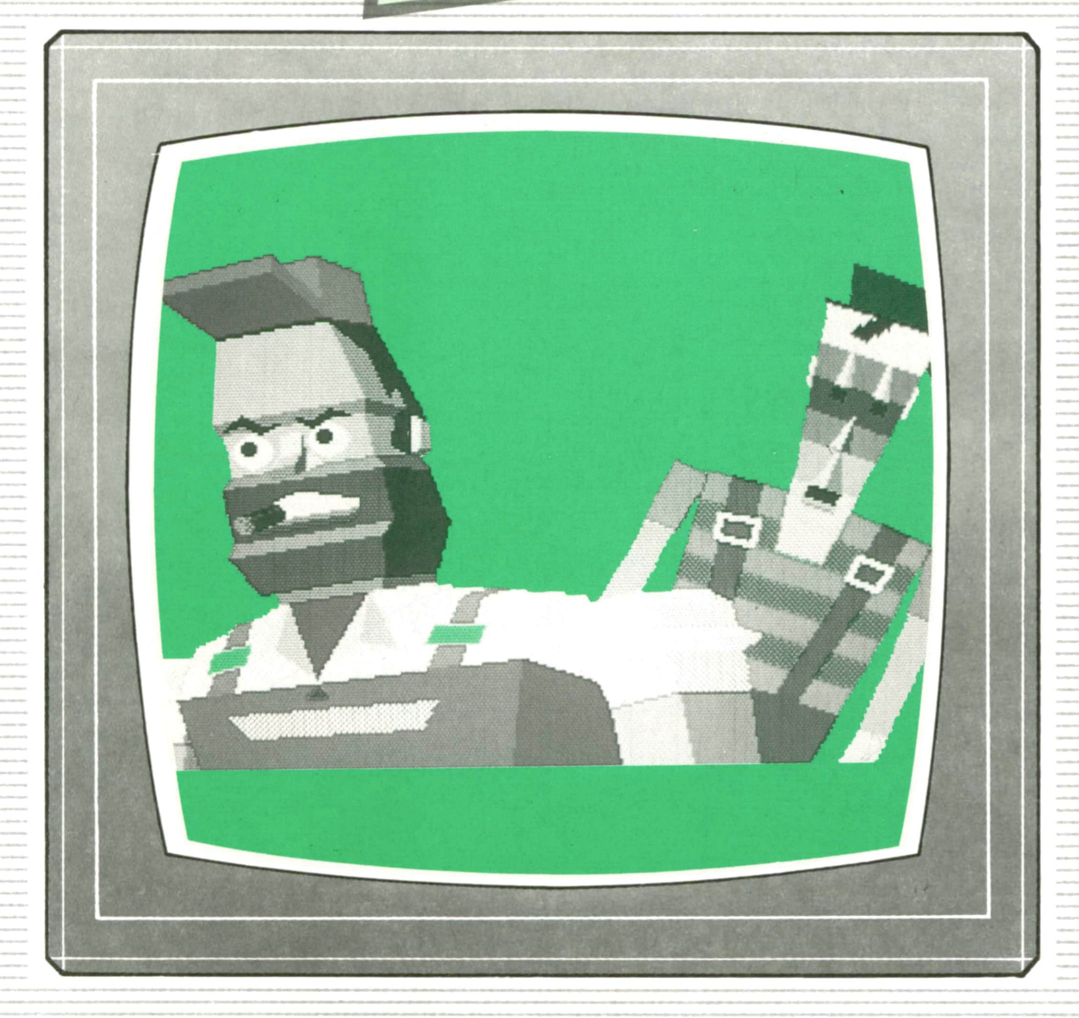
PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Informática y programación 20

PASO A PASO



PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Una publicación de

EDICIONES SIGLO CULTURAL, S.A.

Director-editor:
RICARDO ESPAÑOL CRESPO.
Gerente:
ANTONIO G. CUERPO.
Directora de producción:
MARIA LUISA SUAREZ PEREZ.
Directores de la colección:
MANUEL ALFONSECA, Doctor Ingeniero de Telecomunicación y Licenciado en Informática.
JOSE ARTECHE, Ingeniero de Telecomunicación.
Diseño y maquetación:
BRAVO-LOFISH.
Fotografía:
EQUIPO GALATA.
Dibujos:
JOSE OCHOA

TECNICAS DE PROGRAMACION: Manuel Alfonseca, Doctor Ingeniero de Telecomunicación y Licenciado en Informática. TECNICAS DE ANALISIS: José Arteche, Ingeniero en Telecomunicación. LENGUAJE MAQUINA 8086: Juan Rojas Licenciado en Ciencias Físicas e Ingeniero Industrial. PASCAL: Juan Ignacio Puyol, Ingeniero Industrial. PROGRAMAS (educativos, de utilidad, de gestión y de juegos): Francisco Morales, Técnico en Informática y colaboradores. Coordinador de AULA DE INFORMATICA APLICADA (AIA): Alejandro Marcos, Licenciado en Ciencias Químicas. BASIC: Esther Maldonado, Diplomada en Arquitectura. INFORMATICA BASICA: Virginia Muñoz, Diplomada en Informática. LENGUAJE MAQUINA Z-80: Joaquín Salvachúa, Diplomado en Telecomunicación y José Luis Tojo, Diplomado en Telecomunicación. LENGUAJE MAQUINA 6502: (desde el tomo 5): Juan José Gómez, Licenciado en Química. LOGO: Cristina Manzanera, Licenciada en Informática. APLICACIONES: Sociedad Tamariz, Diplomada en Telecomunicación. OTROS LENGUAJES (COBOL): Eloy Pérez, Licenciado en Informática. Ana Pastor, Licenciada en Informática.

Ediciones Siglo Cultural, S.A.
Dirección, redacción y administración:
Pedro Teixeira, 8, 2.ª planta. Teléf. 810 52 13. 28020 Madrid.
Publicidad:
Gofar Publicidad, S.A. Benito de Castro, 12 bis. 28028 Madrid.
Distribución en España:
COEDIS, S.A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.
Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.
Distribución en Ecuador: Muñoz Hnos.
Distribución en Perú: DISELPESA.
Distribución en Chile: Alfa Ltda.
Importador exclusivo Cono Sur:
CADE, S.R.L. Pasaje Sud América, 1532. Teléf.: 21 24 64.
Buenos Aires - 1.290. Argentina.

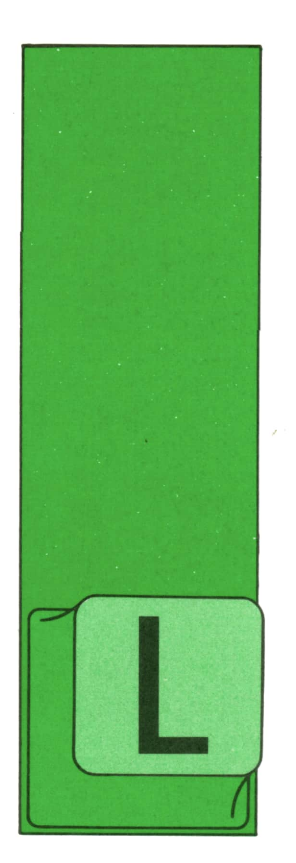
ISBN del tomo: 84-7688-138-X
ISBN de la obra: 84-7688-068-7
Fotocomposición:
ARTECOMP, S.A. Albarracín, 50. 28037 Madrid.
Imprime:
MATEU CROMO. Pinto (Madrid).

Depósito legal: M-5-677-1987
Printed in Spain - Impreso en España.
Suscripciones y números atrasados:
Ediciones Siglo Cultural, S.A.
Pedro Teixeira, 8, 2.ª planta. Teléf. 810 52 13. 28020 Madrid.
Agosto, 1987.
P.V.P. Canarias: 335,–.

INDICE

INFORMATICA BASICA

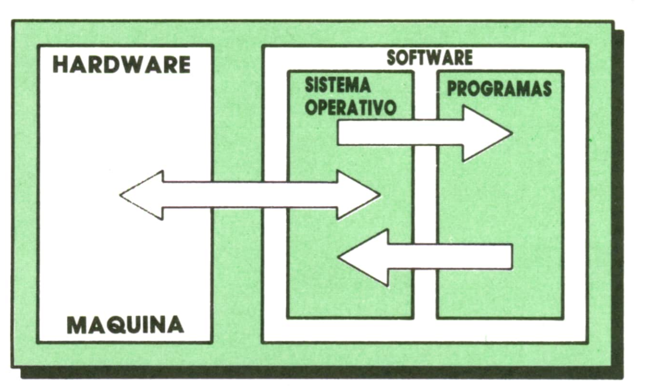
EL MICROPROCESADOR





La unidad central de proceso

LA unidad central de proceso es el núcleo del ordenador. Es, por tanto, un elemento clave en la construcción y desarrollo de cualquier sistema. En nuestro recorrido a través de la CPU, vamos a encontrar, a primera vista, componentes electrónicos y circuitería, comenzando por la tarjeta impresa, que además de servir de soporte, los conecta, constituyendo una limpia red de circuitos integrados, como transistores y puertas lógicas.



La íntima relación que mantienen hardware y software no se ha mantenido en su evolución, siendo la parte mecánica la que ha tenido un desarrollo mucho mayor en los últimos años.

Composición

Podemos diferenciar a grandes rasgos tres componentes fundamentales de la CPU. Son:

— La unidad de control.
— La unidad aritmético-lógica.
— Un conjunto de registros.

Como su nombre indica, la unidad de control (UC) tiene la función de coordinar todos los elementos que van a formar parte de la realización de un proceso concreto. Para ello debe de interpretar correctamente el programa del que recibe las instrucciones, codificarlo convenientemente, y enviar las órdenes oportunas a las unidades implicadas en el proceso. Para realizar esto cuenta con los otros bloques ya citados. Otro componente fundamental es el **reloj,** que se encarga de producir los impulsos necesarios para que las diferentes unidades se sincronicen correctamente.

La unidad aritmético-lógica (ALU) tendrá como misión la ejecución de las operaciones de acuerdo a la estructura interna, ya que no todos los ordenadores realizan los mismos juegos de operaciones. Normalmente el número de operaciones posibles es bastante limitado por lo que será necesario combinarlas para poder realizar los procesos requeridos.

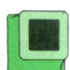
El conjunto de registros que antes mencionábamos constituye una pequeña memoria interna de la que se vale la CPU para manejar correctamente la información precisa en cada momento.

Este conjunto de tres bloques es lo que constituye el **microprocesador.**





Esquema general de la CPU.

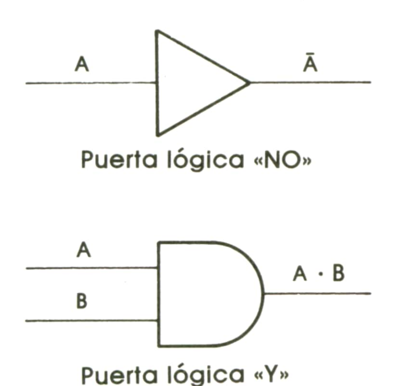


Unidad aritmético-lógica

Los verdaderos componentes del ordenador son los transistores, diodos, resistencias y condensadores. Por tanto, para entender el funcionamiento del ordenador tendremos que considerar varias cosas de las que ya hemos hablado en anteriores capítulos.

— El ordenador utiliza «lógica binaria», es decir, el sistema de numeración en base dos que sólo admite dos dígitos, el 0 y el 1.
— Las funciones lógicas se realizan mediante álgebra de Boole, que utiliza las funciones «y», «o», y «no».

Los elementos físicos que llevan a cabo las funciones lógicas son las llamadas «puertas lógicas». Mediante la combinación de estas puertas se llega a realizar cualquier tipo de función. La función «no» invierte el significado de un elemento. La función lógica «y» solamente es cierta cuando lo son los dos elementos que relaciona. En caso de la función «o», solamente con que se cumpla una de las afirmaciones, la función será válida.



Puerta lógica «NO»

Puerta lógica «Y»



Puerta lógica «O»



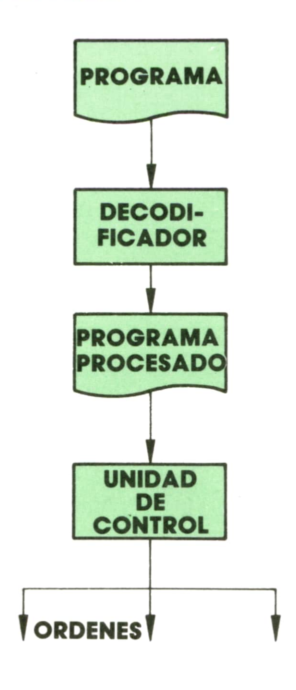
Representación gráfica de las puertas lógicas.



Unidad de control

Como ya se ha comentado, ésta es la parte que se encarga de coordinar todos los elementos adecuadamente para que el ordenador ejecute lo que le pedimos. Evidentemente, esto sólo se puede realizar mediante circuitos de control, con una distribución adecuada del tiempo, que ya hemos dicho que la efectúa el reloj.

La labor de la unidad de control está guiada por el programa, sin embargo, el microprocesador y, por tanto, esta unidad no puede ejecutar sus órdenes directamente. Para ello se dispone de otra pieza electrónica llamada decodificador, que se encarga de «traducir» adecuadamente cada una de las instrucciones del programa a unidades elementales de ejecución que ya tienen significado para el microprocesador.





Ejecución.



Los registros

Son de dos tipos:

— **Generales,** que a su vez están clasificados en dos grupos: principales y alternativos. Pueden utilizarse individualmente o por parejas, ampliando la cantidad de bits que se tratan a la vez. Los registros principales se utilizan como acumuladores auxiliares, y los alternativos sólo los manipulados por instrucciones que intercambian su contenido con el de los principales; se utilizan como lugar seguro donde almacenar los datos temporalmente.

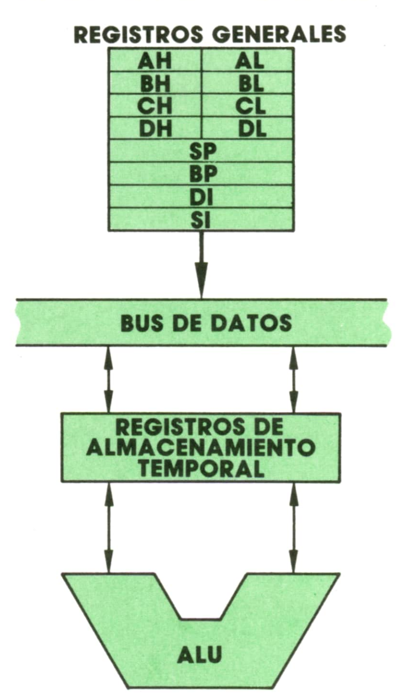
— **Especiales.** Son registros especiales los siguientes:

El **acumulador.** Guarda el resultado de todas las operaciones aritméticas y lógicas.

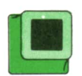
El **registro de estado.** Almacena las señales de condición generadas por la ALU que se utilizarán en las instrucciones de salto; también sirve para dar cuenta de ciertas características, como: acarreo, desbordamientos, paridad, etc.

Contador de programa. Contienen la dirección del byte de la memoria que debe leer el microprocesador.

Puntero de pila. Contiene la dirección actual del tope de la pila, que puede estar colocada en cualquier posición de la memoria externa.



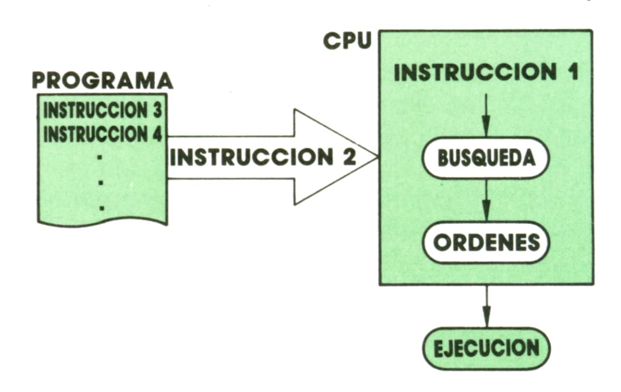

Registros del microprocesador.



Características de un microprocesador

Cada una de las instrucciones del programa, una vez decodificadas, se ejecutará, en una serie de pasos elementales, cada uno de los cuales se realiza en lo que denominamos «ciclo máquina». En cada «ciclo máquina» se efectúan dos tareas; primeramente se identifica el tipo de operación que se pretende realizar (ciclo de búsqueda) y después se pasa a realizar la operación en cuestión (ciclo de ejecución).

La velocidad del microprocesador para realizar una función dependerá de la rapidez con que se ejecuten las instrucciones, es decir, del ciclo máquina, que se mide normalmente en unidades de frecuencia (hertzios: número de ciclos máquina que ocurren en un segundo).




En cada ciclo máquina se identifica la instrucción y se ejecutan las órdenes necesarias.

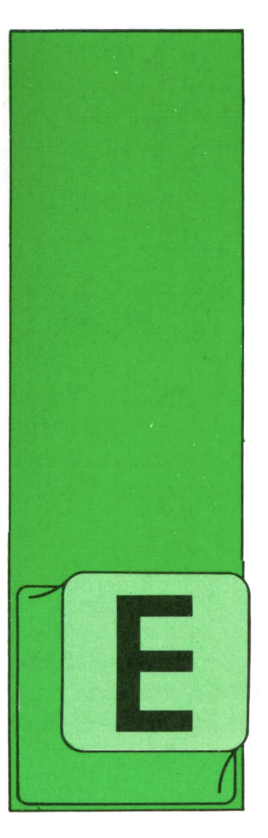
Quizá ésta sea la parte de la informática en la que más se ha evolucionado. Son grandes las diferencias entre los microprocesadores de las primeras y últimas generaciones, sobre todo en las características de los circuitos que los integran: miniaturización, fiabilidad, complejidad y velocidad.

En cuanto a la miniaturización, las diferencias son sorprendentes, de forma que las dimensiones milimétricas que pueden tener hoy ciertos componentes, hace unos años equivalían al volumen de un armario.

El grado de fiabilidad que ofrecen hoy los microprocesadores ha hecho que aumente considerablemente la calidad de funcionamiento de la CPU. La «media de los tiempos de buen funcionamiento» ha pasado de ser de varias decenas de minutos en ordenadores de la primera generación a varios miles de horas para los de la tercera.

En cuanto a complejidad, los circuitos actuales vienen a ser de mil a diez mil veces más complejos.

La velocidad es uno de los puntos más destacables; se ha pasado de CPU's capaces de hacer algunos miles de operaciones por segundo, en las primeras generaciones, a varios millones en los de tercera generación.



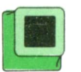
MAQUINA 6502

COMMODORE



Comandos lógicos

ESTOS comandos permiten interrelacionar dos valores, el primero de los cuales debe encontrarse en el ACU, mientras que el segundo se extrae de la memoria según el modo direccionamiento. El resultado se envía al ACU.



Interrelación AND

Se comparan bit a bit el ACU y la posición de memoria direccionada, cuando los dos contengan un «1», también el bit del resultado contendrá un «1». El resto de los casos el resultado será «0».

Si cargamos el ACU con el número $17 y a continuación el comando AND Pt $35 es leído por el microprocesador, se efectuará la siguiente operación

$17 = %00010111
$35 = %00110101
$15 = $00010101

Con el comando AND podrán alterarse los flags Z (Zero) y N (Negativa). Con un resultado de «0» se activará el flag Z, y con un resultado mayor de 127 ($7f) se activará el flag N.



Interrelación OR (inclusivo)

Se relaciona cada bit del ACU con el bit correspondiente del operando, y son posibles los siguientes resultados:

0 ORA 0 = 0
0 ORA 1 = 1
1 ORA 0 = 1
1 ORA 1 = 1

En el momento en que uno de los dos bits esté a «1», el resultado será 1.

La operación del ejemplo anterior sería:

$17 = %00010111
$35 = %00110101
$37 = %00110111

Se activarán los flags N y Z dependiendo del resultado final.



Interrelación OR (exclusivo)

Igual que los dos anteriores, compara bit a bit el contenido del ACU con el de la posición de memoria direccionada, pudiendo presentarse los siguientes casos:

0 FOR 0 = 0
0 FOR 1 = 1
1 FOR 0 = 1
1 FOR 1 = 0

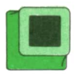
Según el ejemplo propuesto al principio de los comandos lógicos, la interrelación se efectuaría de la siguiente manera:

$17 = %00010111
$35 = %00110101
$22 = %00100010

También aquí se alterarán los flags N y Z.

La siguiente tabla contiene los códigos de comando correspondientes a las tres interrelaciones lógicas que se han expuesto.

Modo de direccionamiento	AND	ORA	EOR
Inmediato	$29	$09	$49
Absoluto	$2D	$0D	$4D
Zero page	$25	$05	$45
Absoluto indexado por X	$3D	$1D	$5D
Absoluto indexado por Y	$39	$19	$59
Zeropage indexado por Y	$35	$15	$55
Indirecto indexado por Y	$31	$11	$51
Indirecto indexado por X	$21	$01	$41



Comando BIT

Este comando, que incluimos dentro del apartado de comandos lógicos, es un poco especial y particular de los procesadores 65 XX.

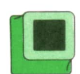
Su acción consiste en efectuar una operación lógica AND entre el ACU y la posición de memoria direccionada, pero sin alterar el contenido de los registros que intervienen.

Si el resultado es cero, el flag Z se activará; en caso contrario, se desactivará. Además, el sexto bit de la posición de memoria direccionada se transfiere al flag V, mientras que el séptimo bit se transfiere al flag N.

De esta manera se pueden controlar los bits 6 y 7 de una posición de memoria y tomar decisiones derivadas de su consulta sin peligro de destrucción del contenido en el registro.

Existen dos modos de direccionamiento para este comando:

Zeropage $24
Absoluto $2C



Comandos de comparación

Con este grupo de comandos podremos hacer comparaciones entre el contenido de los registros del procesador y el contenido de posiciones de memoria utilizando diferentes modos de direccionamiento.

No se alteran los contenidos ni de los registros ni de las posiciones de memoria direccionadas, sino los flags del registro de estado, cuya posterior consulta nos servirá para la toma de decisiones.

Existen comandos para los tres registros de trabajo del procesador:

CMP
CPX
CPY

Comando CMP. Este comando compara el contenido del ACU con el de la posición de memoria direccionada. Para ello se sustrae de aquél el contenido de esta última.

Si se ha producido un desbordamiento negativo, se desactiva el Carry flag; de lo contrario, se activa. Si se obtiene un resultado de «cero», entonces se activa el flag cero, y si el resultado es mayor de 127, se activa el flag Negative.

Imaginemos la siguiente secuencia de instrucciones:

```
LDA # $35
CMP # $43
```

Al realizar la sustracción $35–43$ se obtiene un número negativo menor que cero. Por tanto, tendremos los flags como sigue:

C = 0   Z = 0   N = 1

Probemos ahora otra secuencia diferente:

```
LDA # $45
CMP # $20
```

Ahora la sustracción $45–$20 da como resultado un número mayor que cero y menor de 127 ($7F); por tanto, los flags se alterarían de la siguiente forma:

C = 1   Z = 0   N = 0

El último caso posible sería cuando el ACU y la posición de memoria direccionada o número (direccionamiento inmediato) poseen el mismo valor. Entonces los flags presentarían la configuración siguiente:

C = 1   Z = 0   N = 0

Si estudiamos los tres casos detenidamente, observamos que:

1. El Carry flag se activa para resultados mayores o iguales a cero.
2. El flag Zero se activa si el resultado es igual a cero.
3. El Carry flag se desactiva para un resultado menor que cero.
4. El Carry flag se activa y el flag Zero se desactiva si el resultado es mayor que cero.

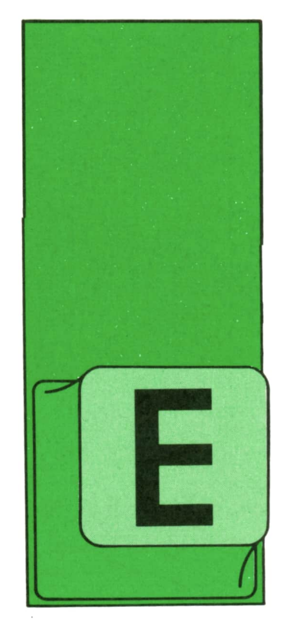
A continuación se presenta una tabla que ayudará cuando se quieran elegir los flags a consultar a la hora de tomar una decisión que dependa del resultado de una de estas comparaciones.

Criterio de decisión	Flags
Mayor o igual	C = 1
Igual	Z = 0
Menor	C = 0
Mayor	C = 1; Z = 0

Comandos CPX y CPY. Análogamente a lo que ocurre con el comando CMP, se opera con estos dos, con la única diferencia de que el registro del que se realiza la sustracción es el registro X o el Y, en vez del ACU.

Para estos comandos, sin embargo, se dispone de menos modos de direccionamiento que en caso de trabajar con el ACU.

Modo de direccionamiento	CMP	CPX	CPY
Inmediato	$C9	$E0	$C0
Absoluto	$CD	$EC	$CC
Zeropage	$C5	$E4	$C4
Absoluto indexado por X	$DD	—	—
Absoluto indexado por Y	$D9	—	—
Zeropage indexado por X	$D5	—	—
Indirecto indexado por Y	$D1	—	—
Indirecto indexado por X	$C1	—	—



PROGRAMAS

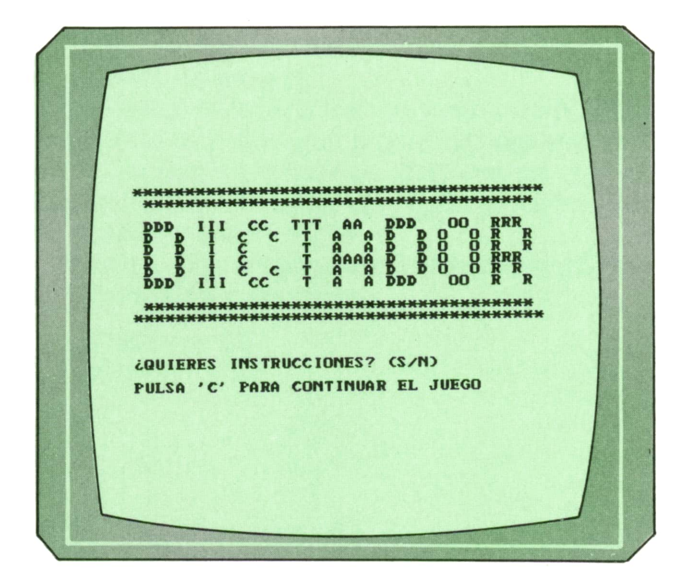
EDUCATIVOS • DE UTILIDAD • DE GESTION • DE JUEGOS



Programa: Dictador

EL programa **Dictador,** del cual damos a continuación el listado, no necesita de ninguna explicación, ya que todas las que se necesitan están incluidas en el programa.

El programa ha sido realizado en un IBM PC bajo GWBASIC, pero puede funcionar sin ningún cambio en el MSX y AMSTRAD. Los usuarios del SPECTRUM sólo tienen que cambiar todas las líneas donde ponga END por GOTO 9999.



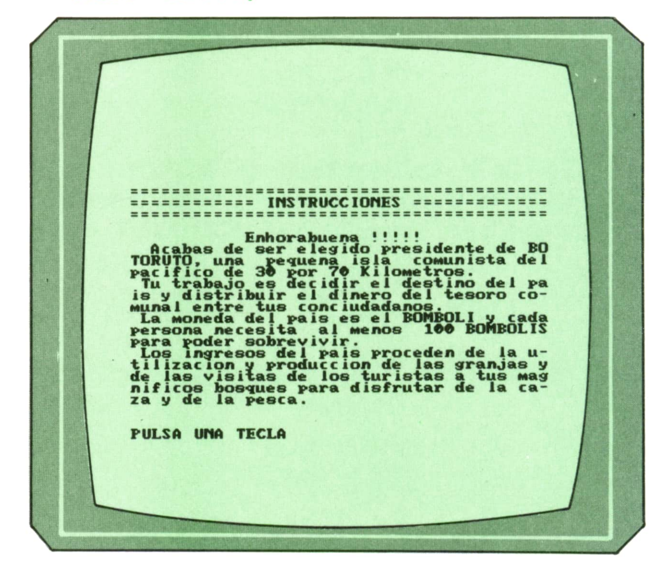


Pantalla de presentación del juego Dictador.

Para los usuarios de COMMODORE se dan las siguientes variaciones:

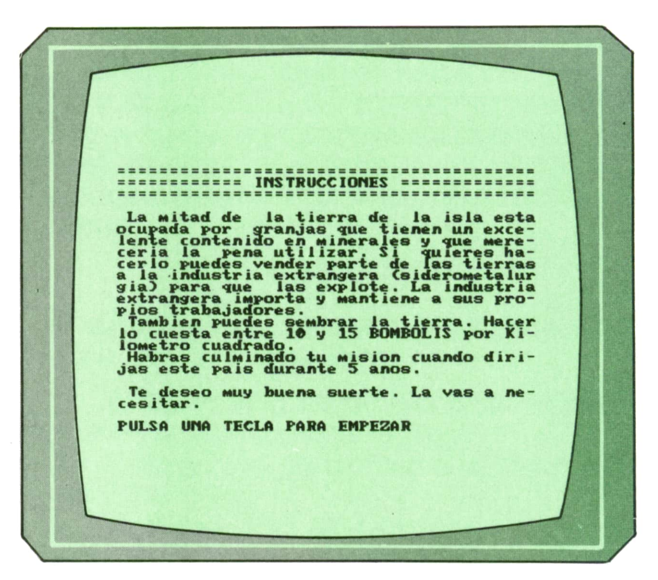
COMMODORE:

```
1170 PRINT CHR$(147)
1340 GET Z$
1400 PRINT CHR$(147)
1620 GET A$
1640 PRINT CHR$(147)
1880 GET A$
1900 PRINT CHR$(147)
2830 PRINT CHR$(147)
2920 PRINT CHR$(147)
4050 GET A$
4450 PRINT CHR$(147)
4560 PRINT CHR$(147)
4860 GET A$
```





Instrucciones del programa.





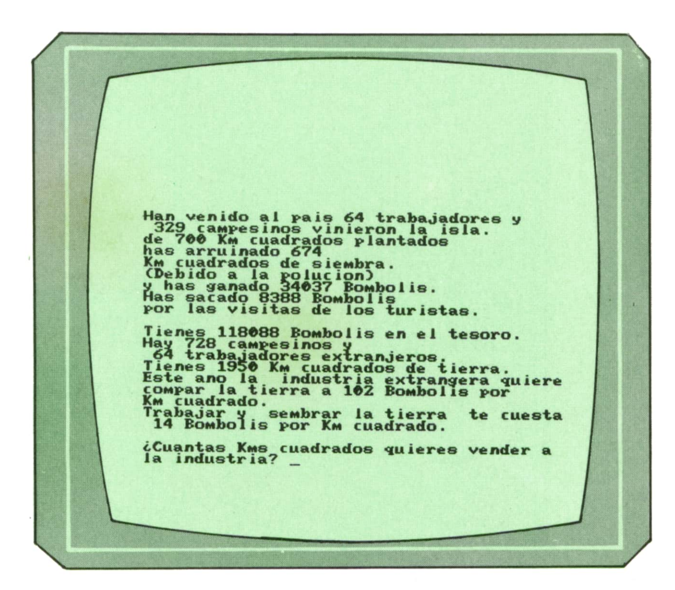
Instrucciones del programa.

```
Tienes 79657 Bombolis en el tesoro.
Hay 399 campesinos y
Tienes 2000 Km cuadrados de tierra.
Este ano la  industria extrangera quiere
compar la tierra a 101 Bombolis por
Km cuadrado.
Trabajar y  sembrar la tierra  te cuesta
 14 Bombolis por Km cuadrado.

¿Cuantas Kms cuadrados quieres vender a
la industria? 50
¿Cuanto dinero distribuyes entre  tus
ciudadanos para poder vivir? 39900
¿Cuantos Km cuadrados quieres cultivar
(recuerda la mano de obra)? 9000

Lo siento, pero cada campesino solo pue-
des cultivar 2 Km cuadrados de tierra.

¿Cuantos Km cuadrados quieres cultivar
(recuerda la mano de obra)?
```





Un momento de la ejecución del programa.

```
1000 REM *****************************************************
1010 REM *                                                   *
1020 REM * EEEE L       DDD  III  CC  TTT  AA  DDD   OO  RRR  *
1030 REM * E    L       D  D  I  C  C  T  A  A D  D O  O R  R *
1040 REM * E    L       D  D  I  C     T  A  A D  D O  O R  R *
1050 REM * EEE  L       D  D  I  C     T  AAAA D  D O  O RRR  *
1060 REM * E    L       D  D  I  C  C  T  A  A D  D O  O R R  *
1070 REM * EEEE LLLL    DDD  III  CC   T  A  A DDD   OO  R  R *
1080 REM *                                                   *
1090 REM *****************************************************
1100 REM
1110 REM
1120 REM ********************************
1130 REM **** (c) Ed. Siglo Cultural ****
1140 REM **** (c) 1987               ****
1150 REM ********************************
1160 REM
1170 CLS
1180 PRINT "*************************************"
1190 PRINT " ***********************************"
1200 PRINT
1210 PRINT " DDD  III  CC  TTT  AA  DDD   OO  RRR"
1220 PRINT " D  D  I  C  C  T  A  A D  D O  O R  R"
1230 PRINT " D  D  I  C     T  A  A D  D O  O R  R"
1240 PRINT " D  D  I  C     T  AAAA D  D O  O RRR"
1250 PRINT " D  D  I  C  C  T  A  A D  D O  O R R"
1260 PRINT " DDD  III  CC   T  A  A DDD   OO  R  R"
1270 PRINT
1280 PRINT " ***********************************"
1290 PRINT "*************************************"
1300 PRINT:PRINT:PRINT
1310 PRINT "(QUIERES INSTRUCCIONES? (S/N)"
1320 PRINT
1330 PRINT "PULSA 'C' PARA CONTINUAR EL JUEGO"
1340 LET Z$=INKEY$
1350 LET N5=INT(RND(1)*4)+4
1360 IF Z$="" THEN 1340
1370 IF Z$="N" OR Z$="n" THEN GOTO 1830
1380 IF Z$="C" OR Z$="c" THEN GOTO 4560
1390 IF Z$<>"S" AND Z$<>"s" THEN GOTO 1340
```

```
1400 CLS
1410 PRINT "========================================"
1420 PRINT "============ INSTRUCCIONES ============="
1430 PRINT "========================================"
1440 PRINT
1450 PRINT "           Enhorabuena !!!!!"
1460 PRINT "  Acabas de ser elegido presidente de BO"
1470 PRINT "TORUTO, una  pequena isla  comunista del"
1480 PRINT "pacifico de 30 por 70 Kilometros."
1490 PRINT " Tu trabajo es decidir el destino del pa"
1500 PRINT "is y distribuir el dinero del tesoro co-"
1510 PRINT "munal entre tus conciudadanos."
1520 PRINT " La moneda del pais es el BOMBOLI y cada"
1530 PRINT "persona necesita  al menos  100 BOMBOLIS"
1540 PRINT "para poder sobrevivir."
1550 PRINT " Los ingresos del pais proceden de la u-"
1560 PRINT "tilizacion y produccion de las granjas y"
1570 PRINT "de las visitas de los turistas a tus mag"
1580 PRINT "nificos bosques para disfrutar de la ca-"
1590 PRINT "za y de la pesca."
1600 PRINT:PRINT
1610 PRINT "PULSA UNA TECLA"
1620 LET A$=INKEY$
1630 IF A$="" THEN GOTO 1620
1640 CLS
1650 PRINT "========================================"
1660 PRINT "============ INSTRUCCIONES ============="
1670 PRINT "========================================"
1680 PRINT
1690 PRINT " La mitad de  la tierra de  la isla esta"
1700 PRINT "ocupada por  granjas que tienen un exce-"
1710 PRINT "lente contenido en minerales y que mere-"
1720 PRINT "ceria la  pena utilizar. Si  quieres ha-"
1730 PRINT "cerlo puedes vender parte de las tierras"
1740 PRINT "a la industria extrangera (siderometalur"
1750 PRINT "gia) para que  las explote. La industria"
1760 PRINT "extrangera importa y mantiene a sus pro-"
1770 PRINT "pios trabajadores."
1780 PRINT " Tambien puedes sembrar la tierra. Hacer"
1790 PRINT "lo cuesta entre 10 y 15 BOMBOLIS por Ki-"
1800 PRINT "lometro cuadrado."
1810 PRINT " Habras culminado tu mision cuando diri-"
1820 PRINT "jas este pais durante";N5;"anos."
1830 PRINT
1840 PRINT " Te deseo muy buena suerte. La vas a ne-"
1850 PRINT "cesitar."
1860 PRINT
1870 PRINT "PULSA UNA TECLA PARA EMPEZAR"
1880 LET A$=INKEY$
1890 IF A$="" THEN GOTO 1880
1900 CLS
1910 LET A=INT(80000! +(1000*RND(1))-(1000*RND(1)))
1920 LET B=INT(400+(10*RND(1))-(10*RND(1)))
1930 LET D=2000
1940 LET W=INT(10*RND(1)+95)
1950 PRINT
1960 PRINT "Tienes";A;"Bombolis en el tesoro."
1970 PRINT "Hay";INT(B);"campesinos y"
1980 LET V9=INT(((RND(1)/2)*10+10))
1990 IF C=0 THEN GOTO 2010
2000 PRINT INT(C);"trabajadores extranjeros."
2010 PRINT "Tienes";INT(D);"Km cuadrados de tierra."
2020 PRINT "Este ano la  industria extrangera quiere"
2030 PRINT "compar la tierra a";W;"Bombolis por"
2040 PRINT "Km cuadradq."
2050 PRINT "Trabajar y  sembrar la tierra  te cuesta"
2060 PRINT V9;"Bombolis por Km cuadrado."
2070 PRINT
```

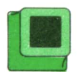
```
2080 PRINT "(Cuantas Kms cuadrados quieres vender a la industria";
2090 INPUT H
2100 IF H<0 THEN GOTO 2080
2110 IF H<D-1000 THEN GOTO 2240
2120 PRINT
2130 PRINT "Piensatelo mejor. solo tienes";D-1000
2140 PRINT
2150 IF X<>0 THEN GOTO 2080
2160 PRINT
2170 PRINT "La industria extranjera solo comprara"
2180 PRINT "tierra de cultivo  porque los bosques"
2190 PRINT "no son rentables para hacer minas de-"
2200 PRINT "bido a los arboles y a las piedas."
2210 PRINT
2220 LET X=1
2230 GOTO 2080
2240 D=INT(D-H)
2250 A=INT(A+(H*W))
2260 PRINT "(Cuanto dinero distribuyes entre  tus"
2270 PRINT "ciudadanos para poder vivir";
2280 INPUT I
2290 IF I<0 THEN GOTO 2260
2300 IF I<A THEN GOTO 2410
2310 IF I=A THEN GOTO 2370
2320 PRINT
2330 PRINT "Intentalo de nuevo. Solo tienes";A
2340 PRINT "Bombolis en el tesoro."
2350 PRINT
2360 GOTO 2260
2370 LET J=0
2380 LET K=0
2390 LET A=0
2400 GOTO 2910
2410 LET A=INT(A-I)
2420 PRINT "(Cuantos Km cuadrados quieres cultivar"
2430 PRINT "(recuerda la mano de obra)";
2440 INPUT J
2450 IF J<0 THEN GOTO 2420
2460 IF J<=B*2 THEN GOTO 2520
2470 PRINT
2480 PRINT "Lo siento, pero cada campesino solo pue-"
2490 PRINT "des cultivar 2 Km cuadrados de tierra."
2500 PRINT
2510 GOTO 2420
2520 IF J<=D-1000 THEN GOTO 2580
2530 PRINT
2540 PRINT "Lo siento, pero solo tienes";D-1000
2550 PRINT "Km cuadrados de tierras de cultivo."
2560 PRINT
2570 GOTO 2420
2580 LET U1=INT(J*V9)
2590 IF U1<A THEN GOTO 2690
2600 IF U1=A THEN GOTO 2660
2610 PRINT
2620 PRINT "No puedes, solo te quedan";A;"Bombolis"
2630 PRINT "en el tesoro."
2640 PRINT
2650 GOTO  2420
2660 LET K=0
2670 LET A=0
2680 GOTO 2910
2690 LET A=A-U1
2700 PRINT "(Cuantos  Bombolis vas a  invertir en el"
2710 PRINT "control de contaminacion";
2720 INPUT K
2730 IF K<0 THEN 2700
2740 IF K<=A THEN 2910
2750 PRINT
```

```
2760 PRINT "Piensalo mejor. Solo tienes";A;"Bombolis."
2770 PRINT
2780 GOTO 2700
2790 IF H<>0 THEN GOTO 2920
2800 IF I<>0 THEN GOTO 2920
2810 IF J<>0 THEN GOTO 2920
2820 IF K<>0 THEN GOTO 2920
2830 CLS
2840 PRINT "Adios. Espero verte pronto."
2850 PRINT:PRINT
2860 PRINT "Si otro dia quieres continuar con el jue"
2870 PRINT "go, pulsa la letra 'C' en la pantalla de"
2880 PRINT "presentacion. Entonces  te hare  algunas"
2890 PRINT "preguntas respecto a tu posicion actual."
2900 END
2910 GOTO 2790
2920 CLS
2930 LET A=INT(A-K)
2940 LET A1=A
2950 IF INT(I/100-B)>=0 THEN GOTO 2980
2960 IF I/100<50 THEN GOTO 4250
2970 PRINT  "Han muerto de hambre";INT(B-(I/100));"ciudadanos"
2980 LET F1=INT(RND(1)*(2000-D))
2990 IF K<25 THEN GOTO 3010
3000 LET F1=INT(F1/(K/25))
3010 IF F1<=0 THEN GOTO 3050
3020 PRINT "Han muerto";F1;"ciudadanos de"
3030 PRINT "inhalacion de monoxido de carbono"
3040 PRINT "y de polvo."
3050 IF INT((I/100)-B)<0 THEN GOTO 3080
3060 IF F1>0 THEN GOTO 3130
3070 GOTO 3240
3080 PRINT "Estas obligado a gastar";INT((F1+(B-(I/100)))*9)
3090 PRINT "Bombolis por gastos de funerales."
3100 LET B5=INT(F1+(B-(I/100)))
3110 LET A=INT(A-((F1+(B-(I/100)))*9))
3120 GOTO 3170
3130 PRINT "Te han obligado a gastar";INT(F1*9)
3140 PRINT "Bombolis por gastos de funerales."
3150 LET B5=F1
3160 LET A=INT(A-(F1*9))
3170 IF A>=0 THEN GOTO 3230
3180 PRINT "No tienes reservas suficientes para"
3190 PRINT "cubrir el coste. Las tierras han te-"
3200 PRINT "nido que ser vendidas."
3210 LET D=INT(D+(A/W))
3220 LET A=0
3230 LET B=INT(B-B5)
3240 IF H=0 THEN GOTO 3290
3250 LET C1=INT(H+(RND(1)*10)-(RND(1)*20))
3260 IF C>0 THEN GOTO 3280
3270 C1=C1+20
3280 PRINT "Han venido al pais";C1;"trabajadores y"
3290 LET P1=INT((((I/100-B)/10)+(K/15)-((2300-D)/50)-(F1/2))/2)
3300 PRINT ABS(P1);"campesinos ";
3310 IF P1<0 THEN GOTO 3340
3320 PRINT "vinieron";
3330 GOTO 3350
3340 PRINT "abandonaron";
3350 PRINT " la isla."
3360 LET B=INT(B+P1)
3370 LET C=INT(C+C1)
3380 LET U2=INT(((2000-D)*((RND(1)+1.5)/2)))
3390 IF C=0 THEN GOTO 3410
3400 PRINT "de";INT(J);"Km cuadrados plantados"
3410 IF J>U2 THEN LET U2=INT(U2/2):GOTO 3430
3420 LET U2=J
3430 PRINT "has arruinado";INT(J-U2);"Km cuadrados de siembra."
```

```
3440 IF U2=0 THEN GOTO 3500
3450 IF T1>=2 THEN GOTO 3500
3460 PRINT "(Debido a";
3470 IF T1=0 THEN GOTO 3490
3480 PRINT "l incremento de";
3490 PRINT " la polucion)"
3500 Q=INT((J-U2)*(W/2))
3510 PRINT "y has ganado";INT(Q);"Bombolis."
3520 LET A=INT(A+Q)
3530 LET V1=INT(((B-P1)*22)+(RND(1)*500))
3540 LET V2=INT((2000-D)*15)
3550 PRINT "Has sacado";ABS(INT(V1-V2));"Bombolis"
3560 PRINT "por las visitas de los turistas."
3570 IF V2=0 THEN GOTO 3660
3580 IF V1-V2>=V3 THEN GOTO 3660
3590 PRINT "El descenso de debe a que"
3600 LET G1=10*RND(1)
3610 IF G1<=2 THEN GOTO 3690
3620 IF G1<=4 THEN GOTO 3720
3630 IF G1<=6 THEN GOTO 3750
3640 IF G1<=8 THEN GOTO 3780
3650 IF G1<=10 THEN 3810
3660 LET V3=INT(A+V3)
3670 LET A=INT(A+V3)
3680 GOTO 3830
3690 PRINT "la pesca ha disminuido porque el"
3700 PRINT "agua esta contaminada."
3710 GOTO 3660
3720 PRINT "la polucion del aire esta matando a"
3730 PRINT "las aves .y no hay caza."
3740 GOTO 3660
3750 PRINT "los banos medicinales estan siendo"
3760 PRINT "arruinados por la polucion del agua."
3770 GOTO 3660
3780 PRINT "un desagradable humo esta estropeando"
3790 PRINT "los banos de sol a los turistas."
3800 GOTO 3660
3810 PRINT "Los hoteles estan sucios por el humo."
3820 GOTO 3660
3830 IF B5>200 THEN GOTO 4080
3840 IF B<343 THEN GOTO 4250
3850 IF (A4/100)>5 THEN GOTO 4320
3860 IF C>B THEN GOTO 3890
3870 IF N5-1=X5 THEN 4450
3880 GOTO 4810
3890 PRINT
3900 PRINT
3910 PRINT "El numero de trabajadores extrangeros"
3920 PRINT "ha excedido al numero de la gente del"
3930 PRINT "propio pais. Como mayoria que son, se"
3940 PRINT "han rebelado y han tomado el pais."
3950 IF RND(1)<=.5 THEN GOTO 4010
3960 PRINT "Te han expulsado de la oficina y ahora"
3970 PRINT "estas viviendo en la carcel. De todas"
3980 PRINT "maneras no te quejes. Otros han acaba-"
3990 PRINT "do muertos."
4000 GOTO 4850
4010 PRINT "Te han asesinado. Lo siento, pero tu ya"
4020 PRINT "no sientes nada."
4030 PRINT
4040 PRINT "PULSA UNA TECLA"
4050 LET A$=INKEY$
4060 IF A$="" THEN GOTO 4050
4070 GOTO 2830
4080 PRINT
4090 PRINT
4100 PRINT B5;"conciudadanos tuyos han muerto en"
4110 PRINT "un solo ano (que exceso). Debido a este"
```

```
4120 PRINT "fracaso tan estrepitoso, no solo te han"
4130 PRINT "derrocado y expulsado de tu oficina sino"
4140 PRINT "que ademas, y para mas inri, te han "
4150 LET M6=INT(RND(1)*10)
4160 IF M6<=3 THEN GOTO 4190
4170 IF M6<=6 THEN GOTO 4210
4180 IF_M6<=10 THEN GOTO 4230
4190 PRINT "sacado el ojo izquierdo a bocados."
4200 GOTO 4020
4210 PRINT "declarado fraude nacional."
4220 GOTO 4020
4230 PRINT "te han hechado a los perros."
4240 GOTO 4020
4250 PRINT
4260 PRINT
4270 PRINT "Alrededor de la mitad de la poblacion"
4280 PRINT "ha muerto desde que tu fuiste elegido"
4290 PRINT "La gente (que queda) te odia (aunque"
4300 PRINT "no mucho. No tienen fuerzas)."
4310 GOTO 3950
4320 IF B5-F1<2 THEN GOTO 3860
4330 PRINT
4340 PRINT "Quedo algun dinero del tesoro que tu"
4350 PRINT "no gastaste. Por ello algunos (mas de"
4360 PRINT "uno) de tus conciudadanos murieron"
4370 PRINT "de hambre. El pueblo esta indignado y"
4380 PRINT "tu estas obligado a dimitir o suicidar-"
4390 PRINT "te (como tu quieras)."
4400 PRINT "La eleccion es tuya"
4410 PRINT "si eliges lo ultimo, por favor apaga el"
4420 PRINT "ordenador antes de hacerlo y no me ensu-"
4430 PRINT "cies mucho. Gracias."
4440 GOTO 4030
4450 CLS
4460 PRINT "FELICIDADES !!!!"
4470 PRINT
4480 PRINT "Has completado con exito tu";N5
4490 PRINT "ano en el poder."
4500 PRINT "Has tenido mucha suerte, pero"
4510 PRINT "sin embargo, se puede decir que"
4520 PRINT "has realizado un buen trabajo."
4530 PRINT "Salud y suerte (probablemente la"
4540 PRINT "necesitaras)."
4550 GOTO 4030
4560 CLS
4570 PRINT "Cuantos anos estubiste en el poder"
4580 PRINT "hasta que te derrocaron";
4590 INPUT X5
4600 IF X5<0 THEN 2830
4610 IF X5<9 THEN 4640
4620 PRINT "Venga. Te crees que soy tonto"
4630 GOTO 4570
4640 PRINT "Cuanto dinero tenias";
4650 INPUT A
4660 IF A<0 THEN 2830
4670 PRINT "Cuantos ciudadanos";
4680 INPUT B
4690 IF B<0 THEN 2830
4700 PRINT "Cuantos trabajadores";
4710 INPUT C
4720 IF C<0 THEN 2830
4730 PRINT "Cuantas Km cuadrados de tierra";
4740 INPUT D
4750 IF D<0 THEN 4020
4760 IF D>2000 THEN 4780
4770 IF D>1000 THEN 1940
4780 PRINT "Venga, empezaste con 1000 Km cuadrados"
4790 PRINT "de cultivo y 1000 Km cuadrados de bosque.
```

```
4800 GOTO 4730
4810 X5=X5+1
4820 B5=0
4830 GOTO 1940
4840 END
4850 PRINT "PULSA UNA TECLA"
4860 LET A$=INKEY$
4870 IF A$="" THEN GOTO 4870
4880 GOTO 2830
```



Generador de SPRITES para COMMODORE

El programa que se propone a continuación nos permitirá definir nuestros propios SPRITES en el COMMODORE para utilizarlos más tarde en nuestros programas. Dichos SPRITES pueden ser almacenados en cinta para que los utilicemos más tarde o para que los modifiquemos cuando queramos.

El programa es bastante autoexplicativo y está lleno de menús para hacer más sencillo su uso al usuario.

Entre las funciones que tiene el programa podemos resaltar:

— Rotación a la derecha.
— Rotación a la izquierda.
— Espejo vertical.
— Espejo horizontal.
— Ampliación en X.
— Ampliación en Y.

El programa está preparado para definir un solo SPRITE cada vez, por lo que se recomienda que, tras su definición, se guarde dicho SPRITE en una cinta de casete mediante el comando SAVE.

```
1000 REM ****************************************
1010 REM * GENERADOR DE SPRITES PARA COMMODORE *
1020 REM ****************************************
1030 REM
1040 REM *****************
1050 REM * INICIALIZACION *
1060 REM *****************
1070 REM
1080 POKE 53280!,1
1090 POKE 53281!,1
1100 POKE 650,128
1110 LET BA=704
1120 LET V=53248!
1130 LET X=0
1140 LET Y=0
1150 FOR I=BA TO BA+62
1160    POKE I,0
1170 NEXT I
1180 POKE V,240
1190 POKE V+1,84
1200 POKE V+23,0
1210 POKE V+29,0
1220 POKE V+39,0
1230 POKE 2040,11
```

```
1240 POKE V+21,1
1250 PRINT CHR$(147);CHR$(145);CHR$(111);
1260 FOR I=1 TO 24
1270    PRINT CHR$(163);
1280 NEXT I
1290 PRINT CHR$(112);CHR$(145);CHR$(145);CHR$(29);"SPRITE";
1300 PRINT CHR$(19);CHR$(145);CHR$(145);
1310 FOR I=1 TO 20
1320    PRINT CHR$(116);SPC(24);CHR$(167)
1330 NEXT I
1340 PRINT CHR$(108);
1350 FOR I=1 TO 24
1360    PRINT CHR$(114);
1370 NEXT I
1380 PRINT CHR$(186);CHR$(17);" C=BORRA.";
1390 FOR I=-1 TO 8
1400    PRINT CHR$(157);
1410 NEXT I
1420 PRINT "H=AYUDA."
1430 GOSUB 2030
1440 LET SP$=""
1450 FOR I=1 TO 26
1460    LET SP$=SP$+" "
1470 NEXT I
1480 REM
1490 REM ***********************
1500 REM * PROGRAMA PRINCIPAL *
1510 REM ***********************
1520 REM
1530 GET A$
1540 IF A$="" THEN GOTO 1530
1550 IF A$="H" THEN GOTO 1740
1560 LET SU=ASC(A$)
1570 IF SU=32 THEN GOSUB 2330
1580 IF SU=67 THEN RUN
1590 LET T1=1105+Y*40+X
1600 IF PEEK(T1)=209 THEN POKE T1,S1:GOTO 1620
1610 POKE T1,46
1620 LET X=X-(SU=80)+(SU=79)
1630 LET Y=Y-(SU=65)+(SU=81)
1640 LET X=X+X*(X>23)-24*(X<0)
1650 LET Y<Y+Y*(Y>20)-21*(Y<0)
1660 GOSUB 2220
1670 IF SU<133 THEN GOTO 1530
1680 PRINT CHR$(19);CHR$(31);CHR$(18);"    EJECUTANDO OPCION    ";CHR$(146)
1690 IF SU>132 AND SU<141 THEN ON SU-132 GOSUB 2440,2800,3260,3620,2620,2940,343
0,3700
1700 PRINT CHR$(19);SP$
1710 GOTO 1530
1720 REM
1730 REM *********************
1740 REM * PANTALLA DE AYUDA *
1750 REM *********************
1760 REM
1770 RESTORE
1780 PRINT CHR$(19);
1790 READ A$
1800 IF A$="*" THEN GOTO 1830
1810 PRINT A$;LEFT$(SP$,26-LEN(A$))
1820 GOTO 1790
1830 GET A$
1840 IF A$="" THEN GOTO 1830
1850 GOTO 1250
1860 DATA " ---PANTALLA DE AYUDA---"
1870 DATA ""
1880 DATA "COMANDOS:"
1890 DATA "F1-ROTACION DERECHA"
1900 DATA "F2-ROTACION IZQUIERDA"
```

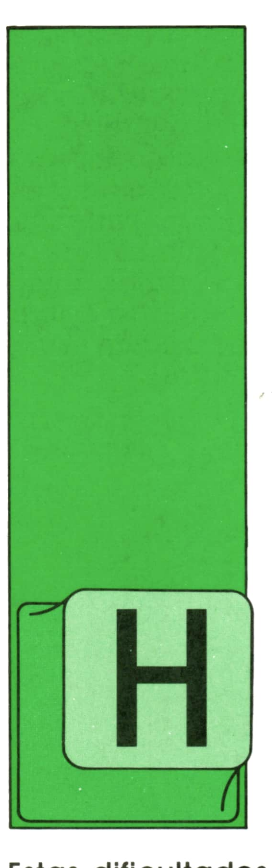
```
1910 DATA "F3-ESPEJO VERTICAL"
1920 DATA "F4-ESPEJO HORIZONTAL"
1930 DATA "F5-SAVE"
1940 DATA "F6-LOAD"
1950 DATA "F7-AMPLIACION X ON/OFF"
1960 DATA "F8-AMPLIACION Y ON/OFF"
1970 DATA ""
1980 DATA "  ---PULSE UNA TECLA---"
1990 DATA "*"
2000 STOP
2010 REM
2020 REM *************************
2030 REM * ESCRITURA DE PARRILLA *
2040 REM *************************
2050 REM
2060 PRINT CHR$(19);CHR$(145);CHR$(145);
2070 FOR I=BA TO BA+62 STEP 3
2080    PRINT CHR$(29);
2090    FOR G=0 TO 2
2100       LET T=128
2110       LET T1=PEEK(I+G)
2120       FOR N=7 TO 0 STEP -1
2130          IF T AND T1 THEN PRINT CHR$(113);:GOTO 2150
2140          PRINT ".";
2150          LET T=T/2
2160       NEXT N
2170    NEXT G
2180    PRINT
2190 NEXT I
2200 REM
2210 REM ************************
2220 REM * ESCRITURA DEL CURSOR *
2230 REM ************************
2240 REM
2250 LET T1=PEEK(BA+Y*3+INT(X/8))
2260 LET T=1105+X+40*Y
2270 LET T2=2^(7-(X AND 7))
2280 IF T2 AND T1 THEN POKE T,209:RETURN
2290 POKE T,174
2300 RETURN
2310 REM
2320 REM ********************
2330 REM * CAMBIO DEL CURSOR *
2340 REM ********************
2350 REM
2360 LET T1=BA+Y*3+INT(X/8)
2370 LET T2=PEEK(T1)
2380 LET T3=2^(7-(X AND 7))
2390 IF T2 AND T3 THEN POKE T1,T2 AND (255-T3):RETURN
2400 POKE T1,T2 OR T3
2410 RETURN
2420 REM
2430 REM *******************
2440 REM * ROTACION DERECHA *
2450 REM *******************
2460 REM
2470 FOR I=BA TO BA+62 STEP 3
2480    LET T1=INT(PEEK(I)/128)
2490    FOR G=I+2 TO I STEP -1
2500       LET T=PEEK(G)
2510       LET T=T*2+T1
2520       LET T1=INT(T/256)
2530       LET T=T AND 255
2540       POKE G,T
2550    NEXT G
2560    POKE I+2,T1 OR PEEK(I+2)
2570 NEXT I
2580 GOSUB 2030
```

```
2590 RETURN
2600 REM
2610 REM **********************
2620 REM * ROTACION IZQUIERDA *
2630 REM **********************
2640 REM
2650 FOR I=BA TO BA+62 STEP 3
2660    LET T1=128*(PEEK(I+2) AND 1)
2670    FOR G=I TO I+2
2680       LET T=PEEK(G)
2690       LET T2=128*(T AND 1)
2700       LET T=INT(T/2)+T1
2710       LET T1=T2
2720       POKE G,T
2730    NEXT G
2740    POKE I,T1 OR PEEK(I)
2750 NEXT I
2760 GOSUB 2030
2770 RETURN
2780 REM
2790 REM *******************
2800 REM * ESPEJO VERTICAL *
2810 REM *******************
2820 REM
2830 FOR I=BA TO BA+2
2840    FOR G=0 TO 30 STEP 3
2850       LET T=PEEK(G+I)
2860       POKE G+I,PEEK(I+60-G)
2870       POKE I+60-G,T
2880    NEXT G
2890 NEXT I
2900 GOSUB 2030
2910 RETURN
2920 REM
2930 REM *********************
2940 REM * ESPEJO HORIZONTAL *
2950 REM *********************
2960 REM
2970 FOR I=BA TO BA+62 STEP 3
2980    LET T$=""
2990    FOR G=I TO I+2
3000       LET T=PEEK(G)
3010       LET T1=128
3020       FOR N=0 TO 7
3030          IF (T AND T1) THEN LET T$=T$+"1":GOTO 3050
3040          LET T$=T$+"0"
3050          LET T1=T1/2
3060       NEXT N
3070    NEXT G
3080    LET C$=""
3090    FOR G=12 TO 1 STEP -1
3100        LET C$=MID$(T$,24-G,1)+C$+MID$(T$,G,1)
3110    NEXT G
3120    FOR G=1 TO 24 STEP 8
3130       LET T=128
3140       LET T1=0
3150       FOR N=0 TO 7
3160          LET T1=T1+T*VAL(MID$(C$,G+N,1))
3170          LET T=T/2
3180       NEXT N
3190       POKE I+INT(G/8),T1
3200    NEXT G
3210 NEXT I
3220 GOSUB 2030
3230 RETURN
3240 REM
3250 REM ********
3260 REM * SAVE *
```

```
3270 REM ********
3280 REM
3290 PRINT CHR$(19);SP$;CHR$(19);
3300 INPUT "NOMBRE = ";N$
3310 IF LEN(N$)>10 OR N$="" THEN GOTO 3290
3320 PRINT CHR$(19);"PULSE UNA TECLA PASA SAVE"
3330 GET A$
3340 IF A$="" THEN GOTO 3330
3350 OPEN 1,1,1,N$
3360 FOR I=BA TO BA+62
3370    PRINT #1,PEEK(I)
3380 NEXT I
3390 CLOSE 1
3400 RETURN
3410 REM
3420 REM ********
3430 REM * LOAD *
3440 REM ********
3450 REM
3460 PRINT CHR$(19);SP$;CHR$(19);
3470 INPUT "NOMBRE = ";N$
3480 IF LEN(N$)>10 OR N$="" THEN GOTO 3660
3490 PRINT CHR$(19);"PULSE UNA TECLA PARA LOAD"
3500 GET A$
3510 IF A$="" THEN GOTO 3500
3520 OPEN 1,1,0,N$
3530 FOR I=0 TO 62
3540    INPUT #1,A
3550    POKE BA+I,A
3560 NEXT I
3570 CLOSE 1
3580 GOSUB 2030
3590 RETURN
3600 REM
3610 REM ***************************
3620 REM * AMPLIACION EN X ON/OFF *
3630 REM ***************************
3640 REM
3650 IF PEEK(V+29)=1 THEN POKE V+29,0:RETURN
3660 POKE V+29,1
3670 RETURN
3680 REM
3690 REM ***************************
3700 REM * AMPLIACION EN Y ON/OFF *
3710 REM ***************************
3720 REM
3730 IF PEEK(V+23)=1 THEN POKE V+23,0:RETURN
3740 POKE V+23,1
3750 RETURN
```

TECNICAS DE ANALISIS

DOCUMENTOS DE SALIDA



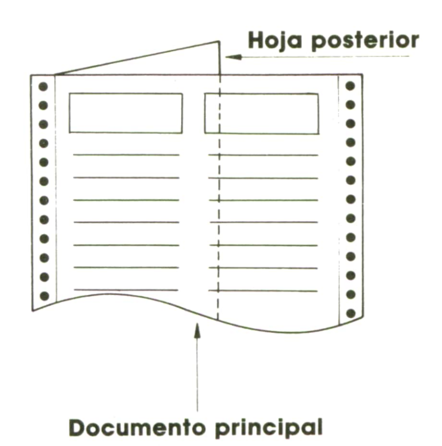
HAY que evitar el tener que dar dos «pasadas» de impresión a un mismo documento, por las dificultades de ajuste que surgen y los errores que se pueden introducir. Estas dificultades llegan a ser casi insalvables si se imprime una misma hoja en papel continuo por ambas caras: piénsese en el caso de que un conjunto de recibos, por ejemplo, se deba imprimir por delante y por detrás y en la secuencia que ha de seguir el ordenador aparece alguna diferencia entre la primera y la segunda «pasada»: a partir del punto en que se produzca este error, ya no coinciden en todo el resto del proceso los datos impresos por el anverso con los del reverso.

f) Es útil tener en cuenta las diversas **manipulaciones o falsificaciones** a que puede estar sometido un documento. Esto es especialmente importante cuando en el impreso aparecen datos delicados, confidenciales o referentes a dinero: algunos documentos impresos por ordenador, en ocasiones, «dan fe» o sirven directamente como «instrumentos de pago». Para ello, hay que marcar las cantidades antes y después de las cifras significativas con asteriscos u otros signos, sin dejar espacios en blanco. En ocasiones, sobre todo si se trata de cantidades, es usual repetir los valores en letra y en número. Si se está utilizando un papel preimpreso, se suelen prever espacios con fondos especiales («aguas» o «anagramas») para evitar que sean borrados o corregidos los datos escritos por la impresora.

g) En ocasiones es muy útil diseñar **hojas laterales, «faldones» o «volantes»** en los preimpresos. Además de la utilidad de control de informaciones a que hemos aludido anteriormente, se pueden utilizar como resguardos de los datos, como resumen que sirva de «recordatorio», como hoja de respuestas a rellenar contestando a las informaciones que se imprimen en el documento, etc. Hay que evaluar, sin embargo, el coste que la impresión de estos papeles especiales puede suponer respecto de la otra opción (sin preimpreso o con él) consistente en escribir varias veces los mismos datos o en preparar varios documentos diferentes (para diferentes usos) a partir de los mismos datos.

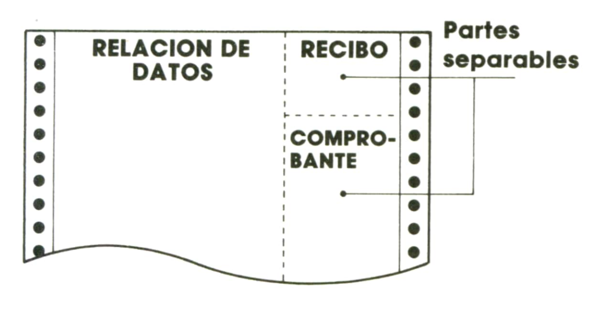
En general, se puede decir que suelen ser de tres tipos, básicamente, los «impresos adicionales» que se suelen diseñar:

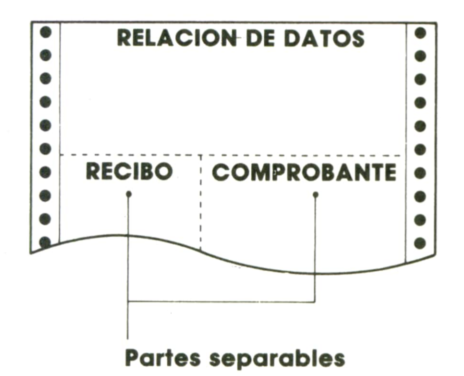
— Como hoja posterior o parcial. En ocasiones se prepara el papel de calco o substancia «autocopiativa» de tal modo que en la «hoja posterior» sólo se escriban parte de los datos que aparecen en la primera hoja. A veces, esta «hoja posterior» sólo ocupa parte de la hoja principal, si el impreso está diseñado para que los datos de la hoja posterior estén en una zona limitada: hay que tener cuidado, en este caso, que el tipo y grosor del papel aseguren el perfecto paso del preimpreso por la impresora.




Preimpreso con hoja posterior.

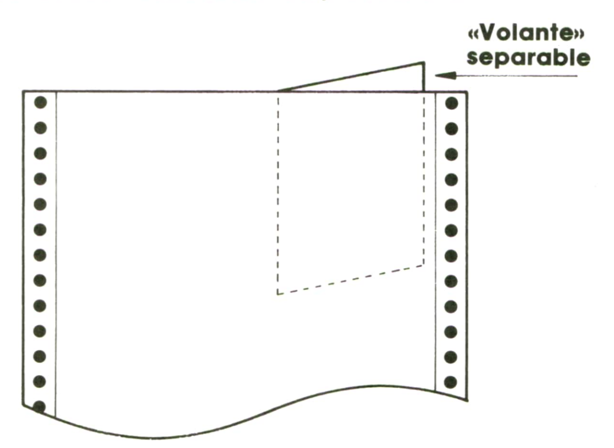
— Como hoja separable. Se puede hacer un «trepado» en el papel para que la parte de la hoja preimpresa pueda ser arrancada o separada del resto del documento. Las partes separables se pueden poner a la derecha de la parte principal del documento si esta parte principal va a ser utilizada como papel en zigzag, o bien en la parte inferior, si el conjunto del documento va a ser manipulado para separar sus hojas. (Ver figura inferior.)





Dos tipos de preimpreso con «partes separables».

— Con copias parciales separables. En ocasiones estos documentos separables se ponen como una pequeña hoja posterior con papel de calco interpuesto o fabricados de papel «autocopiativo». Se suele acudir a esta solución si el contenido del documento no permite «perder» parte de la superficie utilizable y poner hojas laterales (con datos repetidos) para ser arrancadas. En la figura que sigue se muestra la disposición usual de estos «volantes» separables.




Preimpreso con «volante» separable.

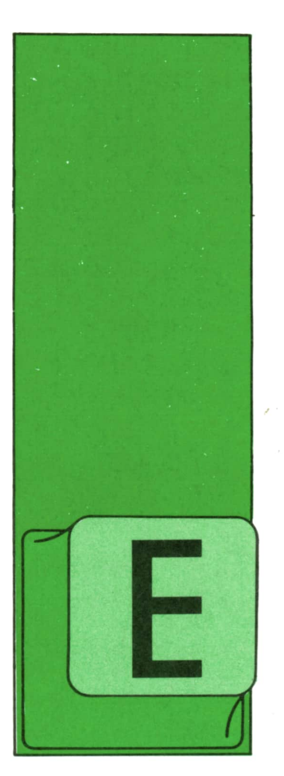
h) Es conveniente, por último, insistir en la importancia de **evaluar los costes** cuando se está diseñando un documento de salida. Ante todo, hay que decidir si se prepara un papel especial preimpreso o se utiliza papel zig-zag (papel «pijama»: hemos comentado los problemas de amortización del papel, de stocks, etc., que conlleva. Por otro lado, y dentro de los primeros, es importante evaluar las variantes posibles (las empresas de fabricación de papel y las imprentas se encargan de que sean muchísimas): con o sin copia, con faldones laterales o volantes, etc.

Además es un factor de importancia el tiempo de impresión, porque la impresora y el ordenador son caros de utilizar.

Hay que elegir incluso, el tipo de papel en función de su utilización y del coste.

Nunca se insistirá suficiente en la necesidad de que el analista tenga en cuenta en sus diseños los elementos externos al «proceso de datos» propiamente dicho: desde el usuario que va a recibir y utilizar el documento hasta las condiciones impuestas por los papeles a utilizar.

TECNICAS DE PROGRAMACION





Instrucciones en bucle (continuación)

EN el lenguaje PASCAL existe también la instrucción MIENTRAS, que tiene una estructura muy semejante a la correspondiente instrucción BASIC:

WHILE condición DO instrucción;

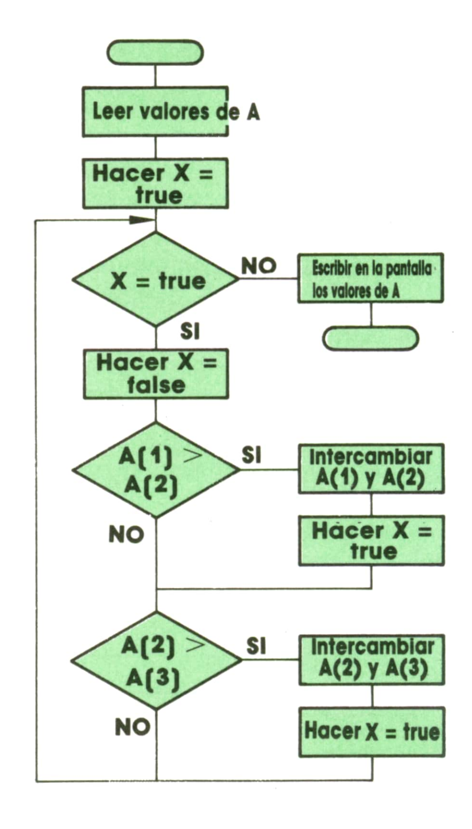
donde «instrucción» puede ser de cualquier tipo: una asignación, un bloque secuencial, una instrucción condicional o un nuevo bucle. Veamos, como ejemplo, cómo se escribe en PASCAL el programa que ordena varios números que vimos en el capítulo anterior en su versión BASIC.

```
program ordenar;
var
   x:boolean;
   y:real;
   a:array[1..3] of real;
begin
   readln (a[1], a[2], a[3]);
   x:=true;
   while x do begin
      x:=false;
      if a[1]>a[2] then begin
         y:=a[2];
         a[2]:=a[1];
         a[1]:=y;
         x:=true;
      end;
      if a[2]>a[3] then begin
         y:=a[3];
         a[3]:=a[2];
         a[2]:=y;
         x:=true;
      end
   end;
   writeln (a[1],' ',a[2],' ',a[3])
end.
```

Observemos algunas diferencias con el mismo programa escrito en BASIC: en primer lugar, utilizamos como variable de control del bucle una variable de tipo booleano (lógico), que es apropiado usar en esta situación. Estas variables pueden definirse en PASCAL, pero no en BASIC, donde tuvimos que recurrir al truco de que la variable pudiera valer únicamente cero o uno.

La segunda diferencia importante es que en BASIC pueden colocarse varias instrucciones entre las palabras reservadas WHILE y WHEN, mientras en PASCAL sólo puede haber (en principio) una instrucción en el bucle WHILE. Sin embargo, utilizando las palabras reservadas BEGIN y END podemos incluir también más de una instrucción dentro del bucle, pues de hecho estamos haciendo uso de un bloque de instrucciones.

El organigrama de este programa PASCAL es casi idéntico al del programa BASIC equivalente:



En cuanto a su ejecución, también es muy parecida:

```
RUN
3 2 1
1.0000000E+000 2.0000000E+000 3.0000000E+000

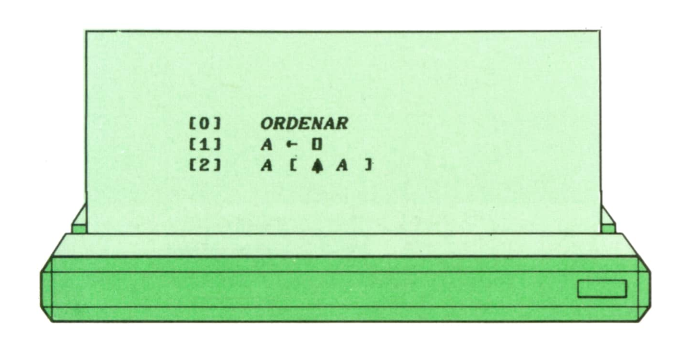
RUN
1 2 3
1.0000000E+000 2.0000000E+000 3.0000000E+000

RUN
1 3 2
1.0000000E+000 2.0000000E+000 3.0000000E+000

RUN
2 1 3
1.0000000E+000 2.0000000E+000 3.0000000E+000
```

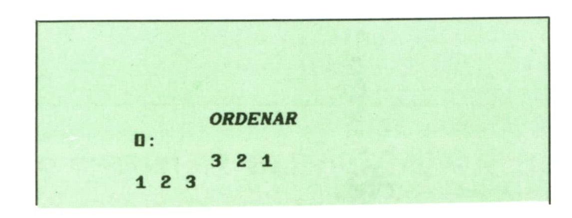
El lenguaje APL no tiene palabras reservadas y carece, por consiguiente, de instrucciones específicas de bucle. Esto no quiere decir que no puedan realizarse bucles de instrucciones. Sin embargo, éstos se construyen por medio de las instrucciones de transferencia condicional, que veremos más adelante. En un capítulo próximo, por tanto, veremos cómo pueden simularse en APL las diversas formas de la instrucción de bucle que veremos a lo largo de este capítulo y los sucesivos.

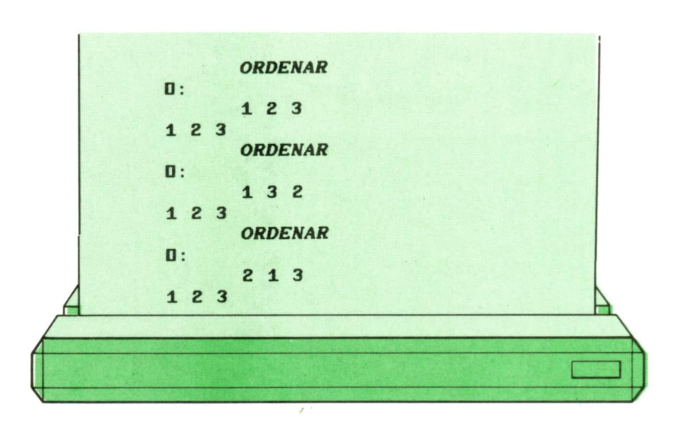
Por otra parte, APL es un lenguaje mucho más potente que cualquiera de los demás, por lo que muchas veces es innecesario construir bucles de instrucciones, pues el algoritmo a realizar puede efectuarse en una sola operación. Este es el caso, precisamente, en el ejemplo que aquí estamos siguiendo (la ordenación de menor a mayor de tres números) que en APL se realiza así:

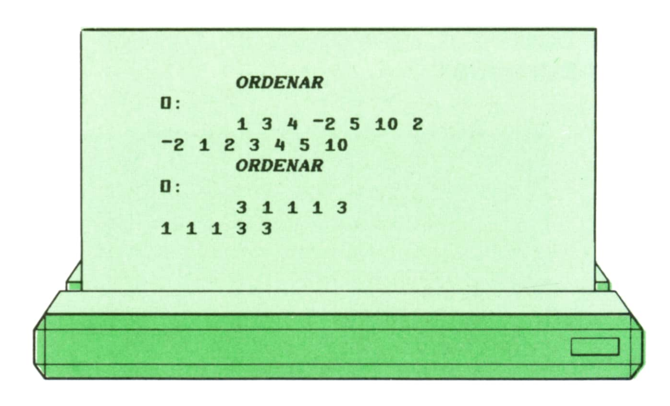


donde la instrucción número 0 define el nombre del programa (ORDENAR, en este caso), la primera instrucción lee del teclado los valores a ordenar y se los asigna a la variable A, y la segunda instrucción los pone en orden de menor a mayor y los escribe en la pantalla. En efecto, la operación representada por un triángulo cruzado por una barra vertical representa una operación APL que obtiene los índices de los elementos de A en orden ascendente. Basta, por tanto, con indexar A por dichos índices para poner los elementos de A en orden ascendente, que es lo que queríamos conseguir, sin necesidad de utilizar bucle alguno. Además, cuando en APL una instrucción realiza una operación, pero no le asigna el resultado a ninguna variable, el resultado aparece automáticamente en la pantalla, por lo que no es necesario añadir una instrucción de escritura al final del programa, como en los casos de BASIC y PASCAL.

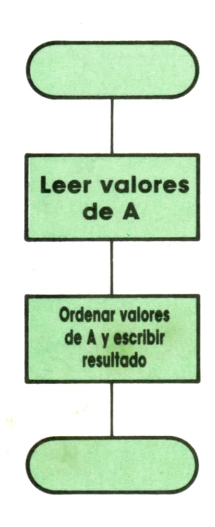
Veamos cómo se ejecutan en APL los ejemplos anteriores:







El organigrama del programa anterior es mucho más sencillo, como es natural, que los de las versiones BASIC y PASCAL:



Existe otra diferencia sustancial entre la versión APL y las versiones BASIC y PASCAL de nuestro programa de ordenación. El programa APL sirve para cualquier número de elementos de A (no sólo tres), mientras que los programas BASIC y PASCAL sólo sirven para ordenar tres elementos. Más adelante veremos cómo pueden generalizarse también, utilizando otras instrucciones de bucle.

Además, el programa APL no tiene necesidad de declarar ninguna de sus variables, por lo que no existirá en principio ningún límite al número de datos que puede ordenar (excepto por las limitaciones intrínsecas de la memoria de la máquina). En BASIC y PASCAL, sin embargo, como veremos más adelante, tendremos siempre que fijar un límite máximo al tamaño de la variable A dentro de nuestro programa.

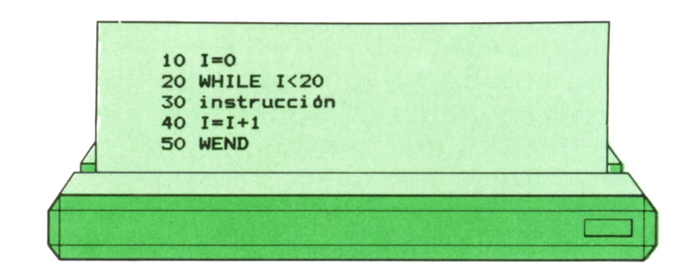
Veamos cómo se aplica el programa APL a la ordenación de un número cualquiera de valores:

Otras instrucciones del bucle

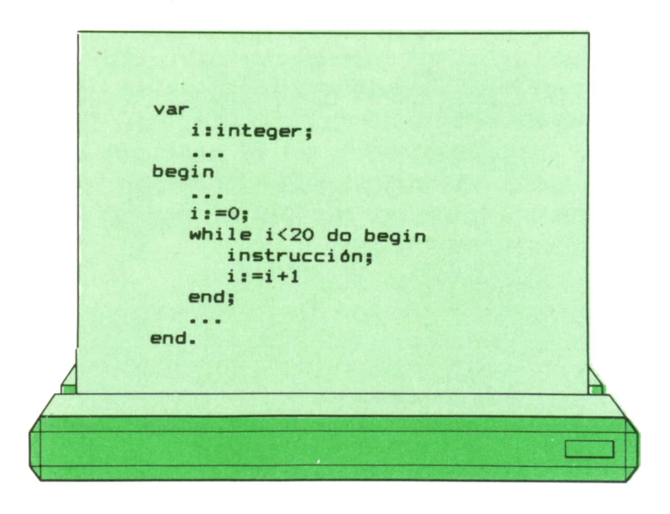
Uno de los casos de bluce más utilizados consiste en ejecutar una o más instrucciones cierto número de veces preespecificado. Esto podría resumirse así:

HACER 20 VECES instrucción;

Es posible realizar esta operación mediante el bucle WHILE que ya hemos visto. En BASIC, por ejemplo, se haría así:



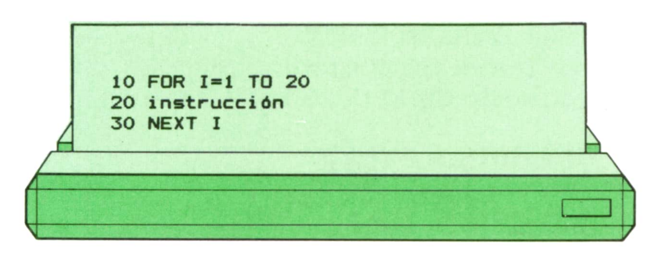
mientras que en PASCAL se podría hacer así:



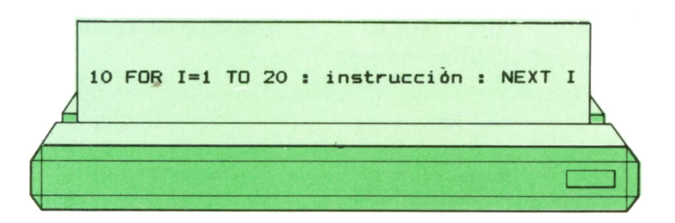
Sin embargo, se trata de una construcción tan frecuente, que casi todos los

lenguajes disponen de una forma reducida (y de palabras reservadas especiales) para definirla. Exceptuamos, como siempre, el lenguaje APL, donde no existe ninguna palabra reservada ni instrucciones especiales de bucle.

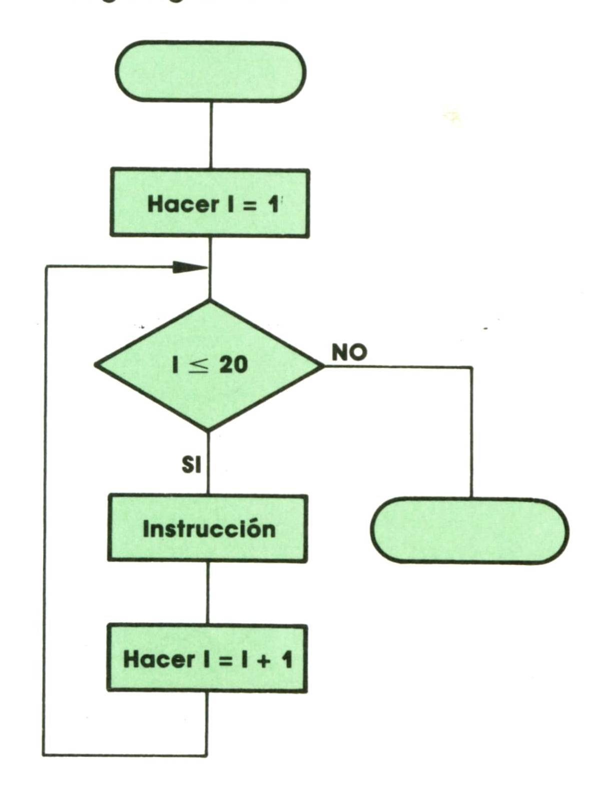
Veamos cómo se construye en BASIC este tipo de estructura:



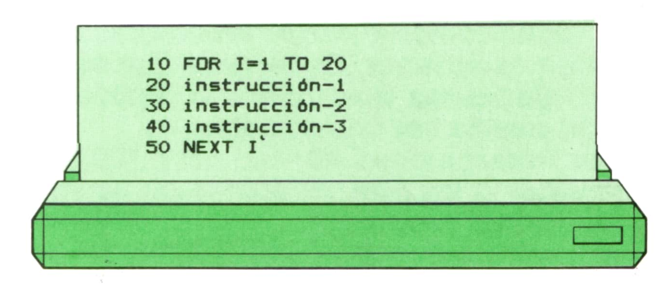
Así, pues, podemos representarla con sólo tres instrucciones, cuya traducción literal es: «Para I = 1 hasta 20, hacer la instrucción y pasar al siguiente valor de I». Como es natural, las tres instrucciones pueden escribirse en una sola línea:



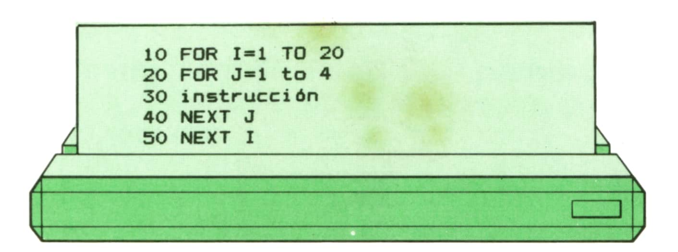
Su organigrama es:



Naturalmente, en lugar de una instrucción podemos incluir varias dentro del bucle:



Estas instrucciones pueden ser asignaciones de valor, instrucciones condicionales u otros bucles, incluso de este mismo tipo. Sin embargo, si se incluye un bucle FOR dentro de otro bucle FOR, es necesario utilizar una variable diferente para cada uno. Además, las instrucciones NEXT tendrán que venir en el orden adecuado: la correspondiente a la variable del bucle más interno debe aparecer primero, tal como en el siguiente ejemplo:



Para terminar, veamos cómo puede utilizarse la instrucción FOR para modificar el programa BASIC que ordena tres números en orden ascendente, de tal manera que pueda aplicarse a la ordenación de un número de valores diferente.

```
10  INPUT N
20  DIM A(N)
30  FOR I=1 TO N : INPUT A(I) : NEXT I
40  LET X=0
50  WHILE X=0
60  LET X=1
70  FOR I=1 TO N-1
80  IF A(I)>A(I+1) THEN LET Y=A(I+1):
    LET A(I+1)=A(I): LET A(I)=Y: LET X=0
90  NEXT I
100 WEND
110 FOR I=1 TO N: PRINT A(I); : NEXT I
```

Veamos lo que hace: la instrucción 10 lee el número de valores que queremos

ordenar, que guardamos en la variable N. La instrucción 20 define la variable A como una serie de N valores.

La instrucción 30 forma un bucle que lee sucesivamente todos los valores que queremos ordenar. Obsérvese que el número de veces que vamos a realizar el bucle puede ser una variable.

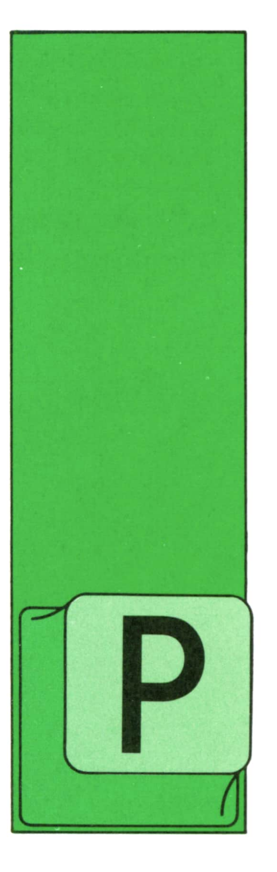
Las instrucciones 40, 50, 60 y 100 son idénticas a las que utilizamos en versión anterior del programa.

Las instrucciones 70, 80, 80 forman un nuevo bucle que compara cada término de la serie con la siguiente y los intercambia en caso de que el primero sea mayor que el segundo (si estaban en orden incorrecto). Este bloque generaliza las dos instrucciones condicionales de la versión anterior al caso de ordenar N valores.

Finalmente, la instrucción 110 utiliza un nuevo bucle para escribir por la pantalla el resultado de la ordenación efectuada.

APLICACIONES

HOJAS DE CALCULO: LOTUS 1-2-3

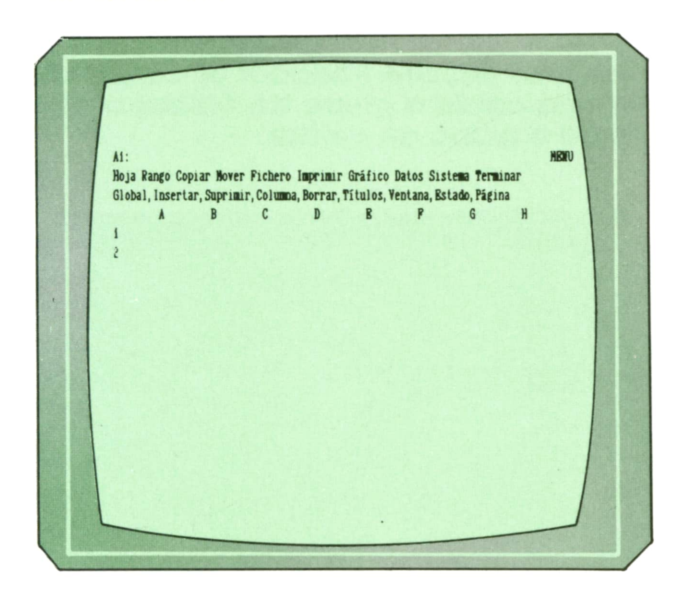




Comandos de la hoja electrónica

PARA conocer cada uno de los mandatos del menú vamos a ir viéndolos en el mismo orden en que aparecen:

HOJA contiene comandos referidos a la hoja de trabajo, presentando las siguientes opciones:



1. GLOBAL. Son mandatos que afectan a la hoja entera:

Formato. Modifica la forma en que se visualizan los valores de una celda o rango de celdas. Los formatos disponibles son: fijo, científico, monetario, general, +/-, %, día, texto y oculta.

Prefijo. Determina si la alineación es a la derecha, en el centro o a la izquierda (por defecto).

Ancho. Varía el ancho de la columna indicada.

Recálculo. Controla el momento, orden y número de veces que se volverán a calcular las fórmulas de la hoja de trabajo cuando se realiza alguna modificación. Podrá ser: natural, por columnas, fila a fila, automático, manual o con iteraciones.

Bloqueo. Funciona con /Rango Proteger y /Rango Dejar para impedir que se modifiquen determinadas celdas de la hoja.

Config. Permite establecer parámetros de configuración en la impresora, directorio, estado...

SuprimeCeros. Determina si en la pantalla aparecerán o no los valores de cero, pero siempre se almacenarán, aunque no aparezcan.

2. INSERTAR. Inserta filas o columnas en la hoja de trabajo, desplazando las ya existentes.

3. SUPRIMIR. Con este comando se pueden eliminar filas o columnas de la hoja, perdiendo así su contenido.

4. COLUMNA. Permite modificar el ancho de una columna.

5. BORRAR. Con este comando desaparece la hoja de trabajo actual y aparece la hoja en blanco inicial. Si no se ha grabado la hoja, se perderá todo su contenido.

6. TITULOS. Este comando fija columnas o filas en una posición de la pantalla para permitir el desplazamiento por una hoja grande. Corresponde a un comando de ventanas en el que pueden señalarse una sola fila o columna.

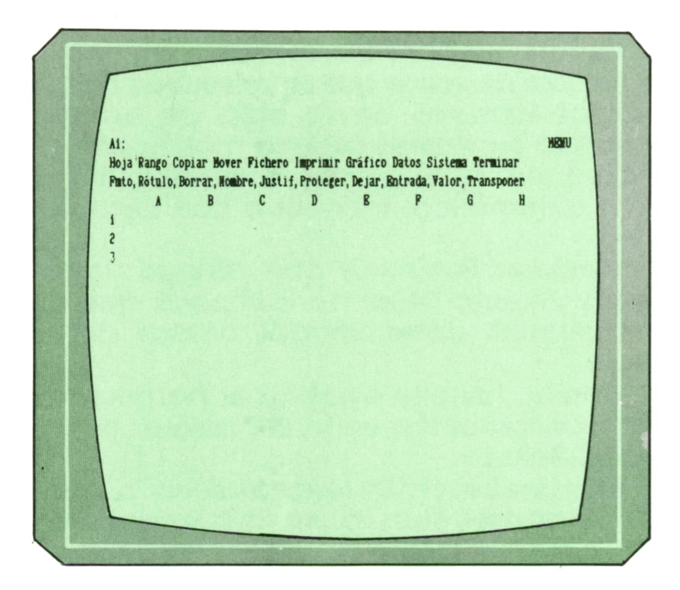
7. VENTANA. Divide la pantalla en dos ventanas horizontales y verticales, se puede pasar de una a otra mediante una

tecla de función. Activando la opción Sinc se sincroniza el movimiento de las dos ventanas simultáneamente.

8. ESTADO. Con este comando se puede visualizar (no modificar) el estado de memoria y especificaciones; pulsando una tecla se vuelve al punto de partida.

9. PAGINA. Este comando sitúa un salto de página en la hoja de trabajo actual de forma que la impresión de la hoja bajo el salto se realiza en la página siguiente.

RANGO permite realizar algunas operaciones con rangos de celdas:



1. Fmto para dar el formato numérico.

2. Rótulo para variar la alineación de los rótulos de la hoja a la derecha, al centro o a la izquierda.

3. BORRAR para eliminar el contenido de una celda o rango de celdas.

4. Nombre para asignar un nombre al rango o a una celda.

5. Justif. Este comando trata texto continuo como un párrafo, cambiando la disposición de las palabras para que ninguna de las líneas sea mayor que el ancho especificado.

6. Proteger para impedir el acceso a una celda o rango.

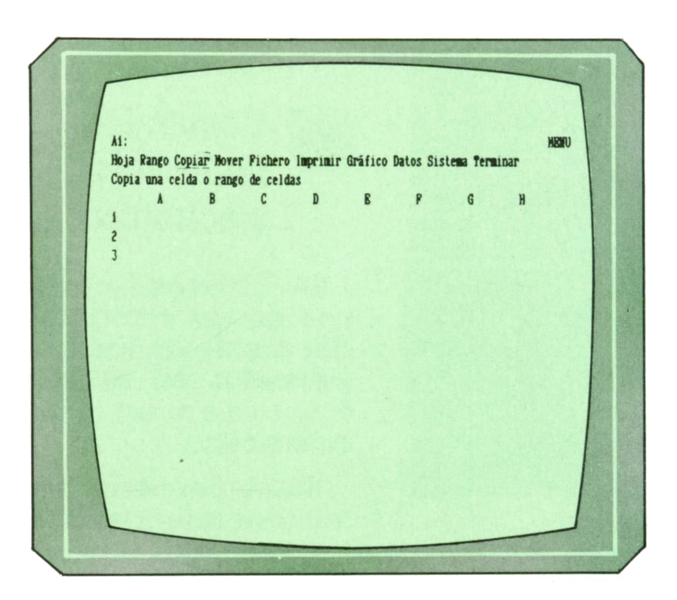
7. Dejar para desproteger celdas.

8. Entrada para limitar el movimiento del cursor a celdas dentro del rango especificado.

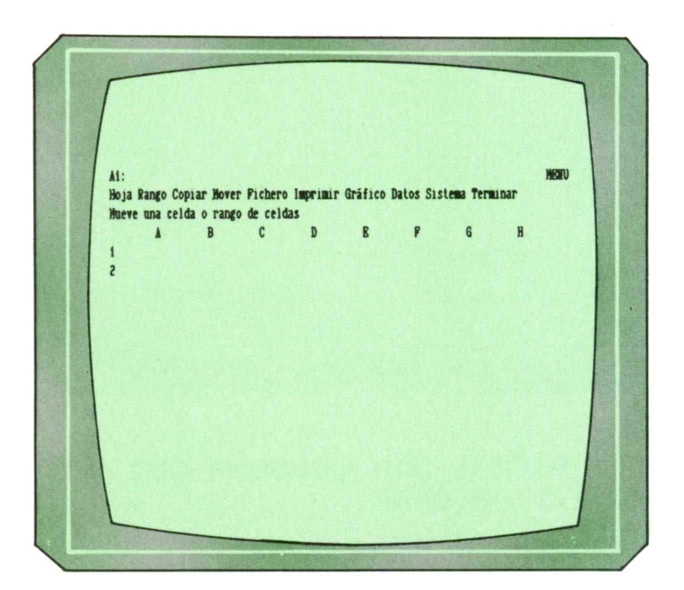
9. Valor convierte una fórmula en su valor.

10. Transponer permite variar la posición de un fila o una columna.

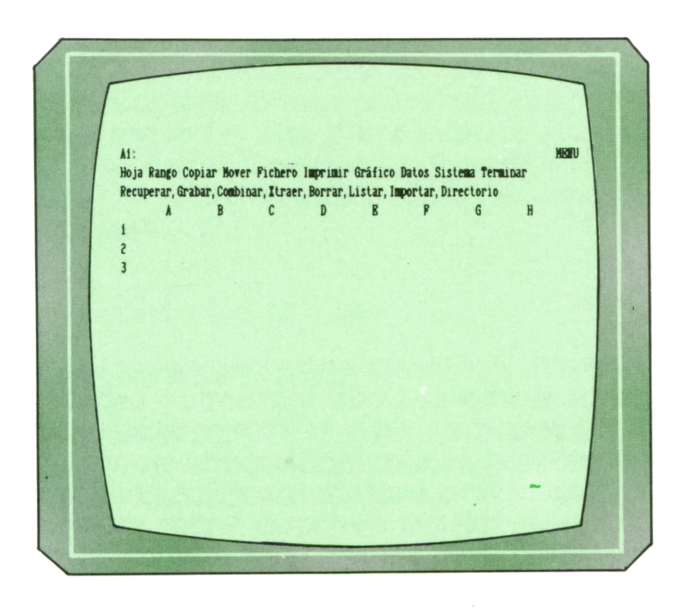
COPIAR permite copiar el contenido de una celda o rango de celdas en otra/s celda/s.



MOVER permite trasladar el contenido de una celda o grupo de celdas a otra celda o grupo de celdas.



FICHERO permite realizar operaciones con ficheros y directorios.



1. Recuperar. Recupera un fichero de tipo hoja del disco y lo copia en la memoria. Los ficheros de hoja llevan la extensión.WK1.

2. Grabar almacena el contenido total de la hoja de trabajo actual en un fichero de tipo .WK1.

3. Combinar incorpora un fichero a la hoja actual en la posición que indique el cursor.

4. Xtraer copia en el fichero seleccionado la hoja o rango de la hoja actual que se indique.

5. Borrar suprime un fichero del disco.

6. Listar presenta en la pantalla una lista de los ficheros que se encuentran en el disco de la unidad activa.

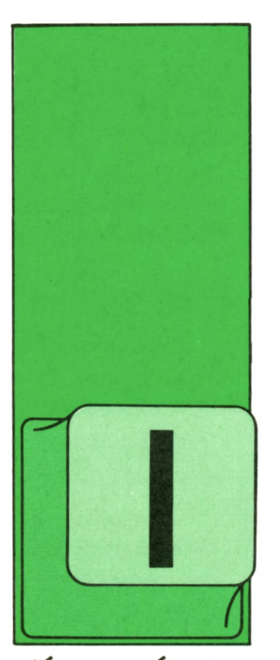
7. Importar incorpora ficheros escritos en ASCII estándar.

8. Directorio permite cambiar el directorio actual de disco.

PASCAL



El tipo SET



MAGINEMOS un programa con un «menú» de opciones numeradas del 1 al 8, de manera que, cuando se escojan las opciones 1, 2, 3 y 4, se tenga que ejecutar una acción común como, por ejemplo, borrar la pantalla; tras la instrucción de lectura de la opción podríamos poner:

```
if (1 <= Opcion) and (Opcion <=4)
then ClrScr;
```

Supongamos ahora que las opciones no fueran éstas, sino, por ejemplo, 1, 3, 4 y 5. Como ya no son consecutivas, para detectar que Opcion es alguna de ellas habría que utilizar un test mucho más complejo que además no valdría si en algún momento cambiásemos las opciones del menú que utilizan el procedimiento:

```
if (Opcion = 1) or
((3 <= Opción) and (Opción <= 5))
then...
```

Si el conjunto de valores de Opcion para el que quisiéramos borrar la pantalla fuese aún mayor y con valores más salteados, la expresión booleana necesaria para indicar si el valor de la variable se encuentra en ese conjunto sería aún más compleja.

Afortunadamente, el PASCAL posibilita trabajar con conjuntos de elementos de una manera muy cómoda y permite, entre otras cosas, comprobar si un dato se encuentra en un conjunto dado directamente.

Los conjuntos formados por 1, 2, 3, 4 y 1, 3, 4, 5, por ejemplo, se expresarían así:

(1,2,3,4), (1,3,2,4) o (4,3,2,1), etc., para el primero y (1,3,4,5) o (1,5,4,3), etc., para el segundo.

es decir, poniendo entre corchetes la lista de elementos que los forman separados por comas; da lo mismo el orden que se utilice. Los elementos pueden ser de cualquier tipo escalar, pero, por supuesto, en un mismo conjunto todos los elementos deben ser del mismo tipo.

Cuando varios de los elementos están seguidos, como es el caso de todos los del primer conjunto y tres del segundo, es posible definirlos de manera abreviada:

(1..4) para el primero y
(1,3..5) o (3..5,1) para el segundo

utilizándose, por tanto, una notación similar a la de los subrangos. El conjunto de todas las letras mayúsculas y minúsculas se podría expresar así:

('A'..'Z', 'a'..'z', 'Ñ', 'ñ')

Antes de entrar en detalles veamos una primera aplicación de los conjuntos que ya hemos mencionado anteriormente: es posible saber si un elemento se encuentra en un conjunto dado por medio del operador IN, que devuelve, según el caso, el resultado lógico TRUE (caso de encontrarse) o FALSE. Con ese operador las dos instrucciones IF anteriores se escribirían de la siguiente manera:

```
if Opción in [1..4]   then ClrScr;
if Opción in [1,3..5] then ClrScr;
```

Es posible definir variables de tipo conjunto, es decir, variables donde poder guardar un conjunto de elementos. Si escribimos:

```
var
  Especiales: set of 1..6;
  DiasLibres: ser of DiaDeLaSemana_t;
```

definiríamos las variables Especiales y DiasLibres, que sirven, respectivamente, para guardar cualquier conjunto de números entre 1 y 6 y cualquier conjunto de días de la semana. O sea, tras las palabras reservadas SET (conjunto en inglés)

y OF se indica el tipo de los elementos que pueden formar parte del conjunto, que debe ser siempre escalar o subrango de éstos. Como sucede con otros tipos, es posible definir antes el tipo de conjunto:

```
type
   Menu_t = set of 1..6;
var
   Especiales: Menu_t;
```

Para guardar conjuntos en una variable se utiliza el operador de asignación de la manera conocida:

```
Especiales := (1..4);
DiasLibres := (Sabado,Domingo);
```

Con estas asignaciones se podrían escribir instrucciones como:

```
if Opción in Especiales then ClrScr;
if not (Hoy in DiasLibres) then
writeln ('Hoy hay que trabajar.');
```

Hoy sería una variable del tipo DiaDeLaSemana_t, que ya conocemos desde hace tiempo. Nótese que se escribe «not (Esto in Aquéllo)», es decir, «not» de una expresión booleana, aunque en lenguaje humano pudiera parecer más lógico escribir «Esto not in Aquéllo».

Dependiendo del compilador, hay un límte para el máximo número de elementos que pueden tener los conjuntos; habitualmente es 256. Por ello, no se podría definir un SET OF INTEGER, pues un conjunto de estas características podría llegar a tener miles de elementos. El mayor conjunto del tipo Menu_t sería, sin embargo (1,2,3,4,5,6), que tiene seis elementos.

En todo caso, el menor conjunto posible es el vacío, aquél que no tiene ningún elemento; se describe simplemente con dos corchetes: ().



Expresiones con conjuntos

Las operaciones entre conjuntos que dan como resultado otro conjunto son la unión, la intersección y la diferencia, que se indican, respectivamente, con los signos +, * y −. Como siempre, en caso de necesidad se pueden utilizar paréntesis.

La unión de dos conjuntos da como resultado otro del mismo tipo formado por los elementos de ambos:

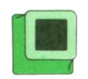
```
('A','E') + ('A','I')  igual a ('A','E','I');
(1) + (2)              igual a (1,2);
( ) + (Lunes)          igual a (Lunes);
```

La intersección, sin embargo, da el conjunto de los elementos comunes a ambos:

```
('A','E') * ('A','I')          igual a ('A');
(1,2) * (3,4)                  igual a ( );
(Lunes) * (Lunes,Martes)       gual a (Lunes);
```

Por último, la diferencia devuelve el conjunto formado por los elementos del primero que no se encuentran en el segundo:

```
('A','E') - ('A','I')          igual a ('E');
(Lunes,Martes)-(Lunes)         igual a (Martes);
```



Operaciones lógicas con conjuntos

Ya conocemos la prueba de pertenencia, que se escribe poniendo en primer lugar el elemento (constante, variable o expresión) cuya pertenencia a un conjunto se desea comprobar, seguida de la palabra reservada IN, tras la que viene el conjunto en cuestión (constante, variable o expresión que dé como resultado un conjunto del tipo adecuado). Formas correctas son:

```
Lunes in (DiasLibres - (Lunes))
1 + 2 in Especiales
```

Entre dos conjuntos se pueden dar las siguientes operaciones lógicas:

— *Igualdad:*

```
(1,2,3)      = ((2,1) + (3))   es TRUE
DiasLibres = ( )               es FALSE
```

— *Desigualdad:*

```
(1,2,3)      <> ((2,1) + (3)) es FALSE
DiasLibres <> ()              es TRUE
```

— *Inclusión,* es decir, comprobar que todos los elementos de uno de ellos están en el otro; se expresa por medio de los operadores que, en otras circunstan-

cias, significan «menor o igual que» (que ahora sería «está incluido en») y «mayor o igual que» («engloba a»):

['B','C'] <= ['A','B','C','D'] es TRUE
['A','B','C','D'] >= ['B','C'] es TRUE
['A'..'Z'] >= ['0'] es FALSE
[] <= DiasLibres es TRUE



Ejemplo

Supongamos que hay que formar equipos de fútbol con los alumnos de una clase y que éstos se encuentran numerados empezando por el 1. Vamos a escribir un programa al que se le faciliten los números de los alumnos de los diferentes equipos que se deseen formar, para que después nos diga qué partidos no se pueden realizar, partiendo de la base de que el número total de alumnos no es un múltiplo exacto de 11 y que, por ello, habrá alumnos figurando en más de un equipo.

Los equipos los guardaremos en variables del tipo «conjunto de alumnos» pero, como puede haber varios, será una tabla de ellas lo que utilizaremos:

```
program Partidos;

const
 Max        = 100;    (* Como máximo 100 alumnos *)
 MaxEquipos = 10;     (* Como máximo 10 equipos  *)

type
 Alumno_t = 1..Max;
 Equipo_t = set of Alumno_t; (* Conjuntos de alumnos *)

var
 Conjunto  : array [1..MaxEquipos] of Equipo_t;

 Total,                (* Para guardar el número de alumnos *)
 NumEquip,             (* Para guardar el número de equipos *)
 Equipo,
 Contrario,
 Alumno    : integer;

(*-------------------------------------------------------------*)
procedure FormarEquipo (var Eq: Equipo_t);
(* Rellena un equipo con los jugadores tecleados *)

  var
   I      : integer;
   Alumno: Alumno_t;
   Ok     : boolean;
 begin
  Eq := [];        (* En principio no tiene jugadores *)
  (*-----------------------------------------------------*)
  (* Pedir los once jugadores e irlos añadiendo al equipo: *)
  (*-----------------------------------------------------*)
  for I := 1 to 11 do
    begin
     repeat
       write ('Alumno: ');
       readln (Alumno);

       (* comprobar que es válido y no está repetido: *)

       Ok := (Alumno in [1..Total]) and not (Alumno in Eq);
       if not Ok then writeln ('No vale. Repita.')

     until Ok;

     Eq := Eq + [Alumno]  (* se añade al equipo *)
    end
 end;

(*-------------------------------------------------------------*)
procedure Comparar ( A, B: Equipo_t);
(* Busca los jugadores comunes a dos equipos *)

  var
   Alumno: Alumno_t;
 begin
  (* busca alumno por alumno: *)
  for Alumno := 1 to Total do
```

```
    if (Alumno in A) and (Alumno in B) then  writeln (Alumno)
 end;

(*-------------------------------------------------------------*)
procedure MostrarEquipos (Al: Alumno_t);
(* Busca los equipos en los que está un alumno *)

  var
   Equipo      : integer;
   TieneEquipo: boolean;
 begin
  write ('Alumno',Al:3,'  Equipos: ');

  (* todavía desconocemos si tiene equipo: *)
  TieneEquipo := false;

  (* para cada alumno, busca equipo por equipo: *)
  for Equipo := 1 to NumEquip do
    if Al in Conjunto [Equipo] then
      begin
       TieneEquipo := true;
       write (Equipo,' ')
      end;

  if TieneEquipo then writeln
                else writeln ('No figura en ninguno.')
 end;

(*-------------------------------------------------------------*)
begin
 ClrSCr; (* o PAGE, etc. *)
 write ('Número total de alumnos: ');
 readln (Total);
 write ('Número de equipos: ');
 readln (NumEquip);

 (*-----------------------------------*)
 (* Formar los equipos de uno en uno: *)
 (*-----------------------------------*)
 for Equipo := 1 to NumEquip do
   begin
    writeln;
    writeln ('Equipo número ',Equipo);
    FormarEquipo (Conjunto [Equipo])
   end;

 (*------------------------------------------*)
 (* Probar todos los emparejamientos posibles: *)
 (* (véase la explicación en el texto)       *)
 (*------------------------------------------*)
 writeln;
 for Equipo := 1 to NumEquip - 1 do
  for Contrario := Equipo + 1 to NumEquip do

     (* mirar si hay jugadores compartidos: *)

     if Conjunto [Equipo] * Conjunto [Contrario] <> [] then
       begin
        writeln;
        write   ('No puede jugar el equipo ',Equipo);
        writeln (' contra el equipo ',Contrario);
        writeln ('Los jugadores comunes son: ');

        Comparar (Conjunto [Equipo], Conjunto [Contrario])
       end;

 (*-------------------------------------*)
 (* Mostrar los equipos de cada alumno: *)
 (*-------------------------------------*)
 writeln ('Pulse Intro.');
 readln;
 writeln;
 writeln ('Listado por alumnos.');
 writeln ('--------------------');
 for Alumno := 1 to Total do MostrarEquipos (Alumno)

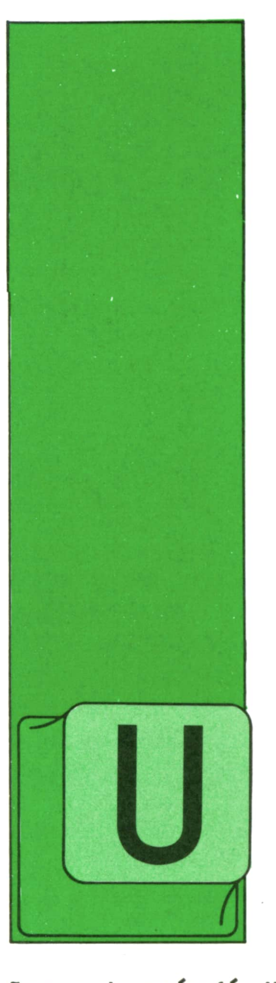
end.
```

Supongamos que hubiera cuatro equipos. En este caso, los partidos posibles (en principio) serían:

— el 1 contra los equipos 2, 3 y 4.
— el 2 contra los equipos 3 y 4.
— el 3 contra el equipo 4.

y de ahí los dos bucles FOR utilizados para ver los partidos posibles; por otra parte, la condición para que un partido se pueda celebrar es que los dos equipos no tengan jugadores comunes.

OTROS LENGUAJES
COBOL





Procedimientos

UNA de las últimas técnicas de programación aboga por la modularización de los programas. Consiste en descomponer un gran problema en otros más pequeños y de más fácil solución.

También ocurre con relativa frecuencia que determinado grupo de instrucciones se ejecutan varias veces en un mismo programa.

Para realizar la modularización y evitar que un mismo grupo de instrucciones aparezcan más de una vez, existen los procedimientos.

El formato de esta instrucción es:

PERFORM nombre-procedimiento-1 (THRU nombre-procedimiento-2)

Si en una sentencia PERFORM sólo se pone un nombre de párrafo, se ejecutan las instrucciones que lo formen y cuando acabe con la última, retorna a la instrucción que sigue al PERFORM. Es importante tener claro el concepto de retorno: siempre se continúa ejecutando la instrucción que se encuentra a continuación del PERFORM.

Si se desea ejecutar un grupo de procedimientos, se pone el nombre del párrafo del primero de ellos en nombre-procedimiento-1 y el último en nombre-procedimiento-2.

```
IDENTIFICATION DIVISION.
 PROGRAM-ID. EJ-PERFORM.

ENVIRONMENT DIVISION.

DATA DIVISION.

 WORKING-STORAGE SECTION.

  01 DATO-1          PIC 9(3).
  01 DATO-2          PIC 9(3).

PROCEDURE DIVISION.

INICIO.
        DISPLAY 'TECLEE PRIMER DATO'.
        ACCEPT DATO-1.
        DISPLAY 'TECLEE SEGUNDO DATO'
        ACCEPT DATO-2.
        PERFORM OPER-2.
        DISPLAY DATO-1 ' ' DATO-2.
        PERFORM OPER-1 THRU OPER-3.
        DISPLAY DATO-1 ' ' DATO-2.

FIN-PROGRAMA.
        STOP RUN.

OPER-1.
        ADD 10 TO DATO-1.
        IF DATO-2 NOT < 10
           SUBTRACT 10 FROM DATO-2.

OPER-2.
        ADD 20 TO DATO-1.
        IF DATO-2 NOT < 20
           SUBTRACT 20 FROM DATO-2.

OPER-3.
        ADD 30 TO DATO-1.
        IF DATO-2 NOT < 30
           SUBTRACT 30 FROM DATO-2.
```

En este programa se observa cómo los procedimientos pueden aparecer a continuación del STOP RUN, ya que éstos no se ejecutan en línea al ser llamados desde un PERFORM.

Suponiendo que los valores introducidos a DATO-1 y DATO-2 son 940 y 50, respectivamente, la ejecución del programa sería la siguiente:

En primer lugar, se llama al procedimiento OPER-2. Por tanto, se sumará 20 a DATO-1 y como DATO-2 es mayor que 20 se les restará 20. En este punto se ha finalizado la ejecución del procedimiento OPER-2 y se retorna a la instrucción siguiente: DISPLAY DATO-1 '' DATO-2.

En pantalla se presenta 960 030.

El siguiente PERFORM establece que deben ejecutarse los procedimientos OPER-1, OPER-3 y todos los que se encuentren entre ambos. Se ejecutan en secuencia OPER-1, OPER-2 y se llega a OPER-3. La instrucción de retorno displaya en la pantalla el resultado de la ejecución: 020 000.

El programa finaliza con el STOP RUN.



Iteraciones

Todos los ejemplos vistos hasta ahora se limitan a tomar unos datos, tratarlos y escribir el resultado de las operaciones una sola vez. Pero lo normal sería que el programa continuase pidiendo datos hasta que el programador desease finalizar, por lo que se deben poder establecer repeticiones o iteraciones. Para ello existe un formato nuevo de la misma instrucción PERFORM.

PERFORM nombre-procedimiento-1 (THRU nombre-procedimiento-2) UNTIL condición.

El procedimiento o rango de procedimientos seleccionados se ejecutará mientras la condición que acompaña al PERFORM no se cumpla. Se ejecuta la siguiente instrucción sólo cuando la condición sea cierta.

Como ejemplo pensemos en una tienda que desea obtener al final del día el total de las ventas obtenidas.

```
IDENTIFICATION DIVISION.
 PROGRAM-ID. EJ-ITERACIONES.

ENVIRONMENT DIVISION.

DATA DIVISION.

 WORKING-STORAGE SECTION.

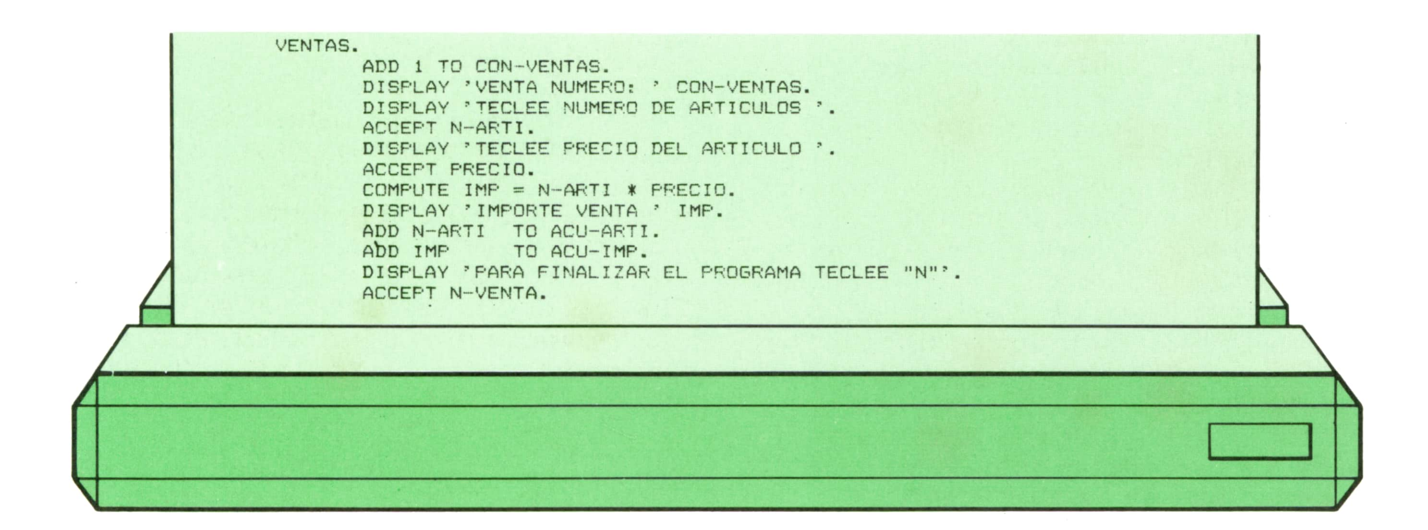
   01 N-VENTA              PIC X.
   01 CON-VENTAS           PIC 9(3)       VALUE ZERO.
   01 ACU-ARTI             PIC 9(5)       VALUE ZERO.
   01 ACU-IMP              PIC 9(8)       VALUE ZERO.
   01 IMP                  PIC 9(5).
   01 MEDIA-ARTI           PIC 9(2).
   01 MEDIA-IMP            PIC 9(5).

   01 DATOS-VENTA.
      05 N-ARTI            PIC 9(2).
      05 PRECIO            PIC 9(3).

PROCEDURE DIVISION.

INICIO.
        DISPLAY 'PARA FINALIZÁR EL PROGRAMA TECLEE "N"'.
        ACCEPT N-VENTA.
        PERFORM VENTAS UNTIL N-VENTA = 'N'.
        IF CON-VENTAS > 0
           COMPUTE MEDIA-ARTI = ACU-ARTI / CON-VENTAS
           COMPUTE MEDIA-IMP  = ACU-IMP / CON-VENTAS
        ELSE
           MOVE ZERO TO MEDIA-ARTI
           MOVE ZERO TO MEDIA-IMP.
        DISPLAY 'ARTICULOS VENDIDOS        ' ACU-ARTI.
        DISPLAY 'MEDIA ARTICULOS VENDIDOS  ' MEDIA-ARTI.
        DISPLAY 'IMPORTE DE LAS VENTAS     ' ACU-IMP.
        DISPLAY 'MEDIA IMPORTE DE VENTAS   ' MEDIA-IMP.

FIN-PROGRAMA.
        STOP RUN.
```



```
VENTAS.
        ADD 1 TO CON-VENTAS.
        DISPLAY 'VENTA NUMERO: ' CON-VENTAS.
        DISPLAY 'TECLEE NUMERO DE ARTICULOS '.
        ACCEPT N-ARTI.
        DISPLAY 'TECLEE PRECIO DEL ARTICULO '.
        ACCEPT PRECIO.
        COMPUTE IMP = N-ARTI * PRECIO.
        DISPLAY 'IMPORTE VENTA ' IMP.
        ADD N-ARTI  TO ACU-ARTI.
        ADD IMP     TO ACU-IMP.
        DISPLAY 'PARA FINALIZAR EL PROGRAMA TECLEE "N"'.
        ACCEPT N-VENTA.
```

Los campos CON-VENTAS, ACU-ARTI y ACU-IMP van a ser acumuladores, se les irán sumando cantidades, por lo que deben tener un valor inicial para evitar los errores de ejecución.

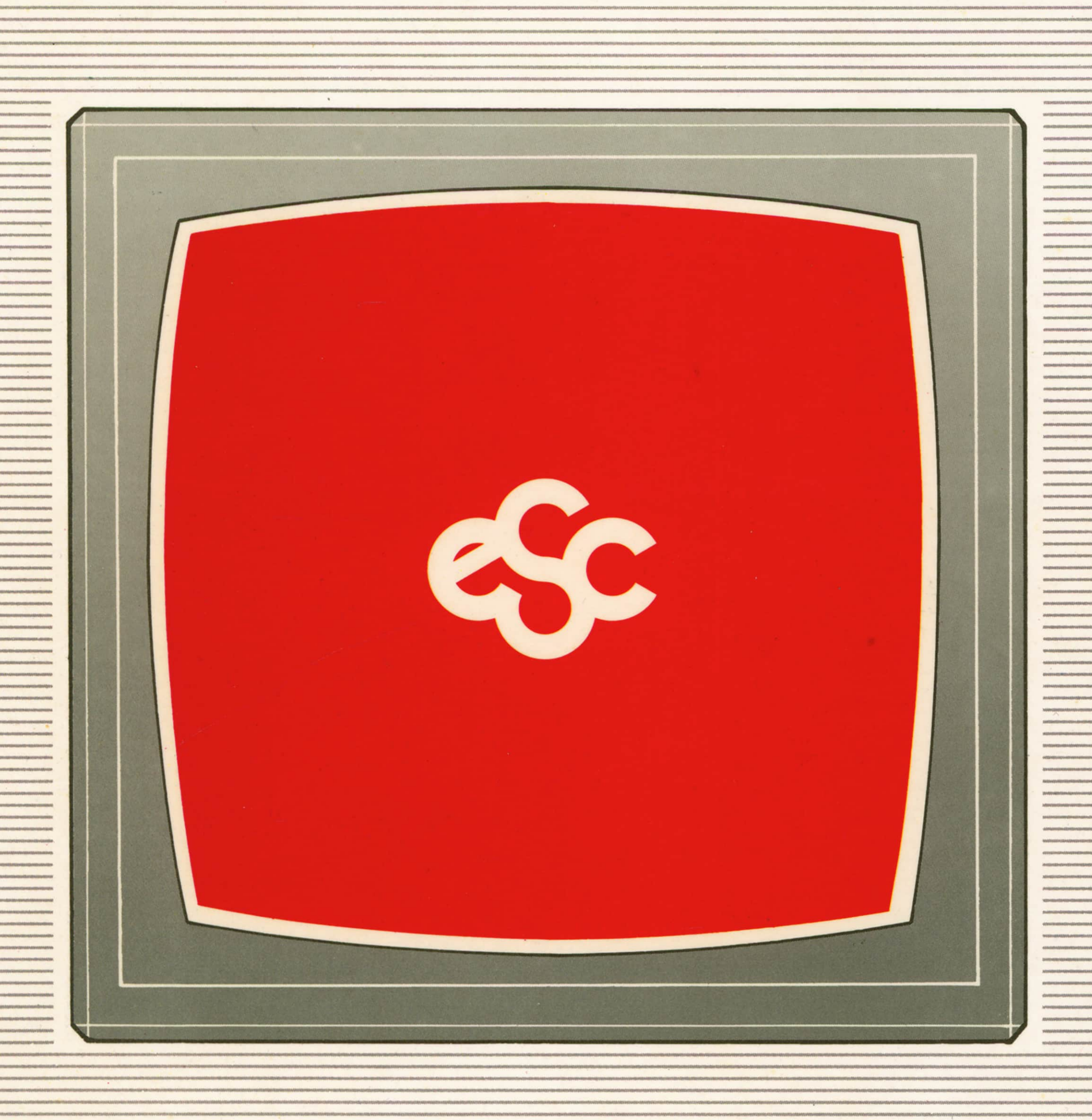
En primer lugar, se pregunta si se desea terminar el programa. Si el contenido de N-VENTA es distinto de «N», la condición de UNTIL es falsa y las instrucciones del procedimiento se ejecutan.

En el procedimiento VENTAS se incrementa en uno el valor de CON-VENTAS (número de ventas realizadas).

Se pide el resto de los datos, calculando el importe de la venta, acumulando éste. Al final del procedimiento se vuelve a preguntar si se desea finalizar.

Cuando N-VENTA sea «N», no se ejecutará el procedimiento VENTAS y sí el IF que le sigue. En este IF se comprueba si se ha realizado algna venta con el fin de no provocar un error de ejecución si el número de ventas es cero.

Antes de finalizar se muestran los resúmenes pedidos.



▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼